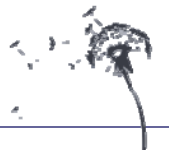
SemEUsE

SEMantiquE pour bUS de sErvice

2007 TLOG 018

Document name:	Integration of security services and security patterns in service composition
Document version:	1.0
Task code:	T3.3
Deliverable code:	T3.3D1
WP Leader (organisation):	FT
Deliverable Leader (organisation):	INRIA ARLES
Authors (organisations):	INRIA ARLES, INSA, EBM, FT, INT, TCF, LIP6
Date of first version:	27/09/2009



Change control

Changes	Author / Entity	Code of version
Creation of the document	Nikolaos Georgantas (INRIA ARLES)	V0.1, 24/07/2009
Integration of partners contributions	Nebil Ben Mabrouk (INRIA ARLES), Sandrine Beauche (INRIA ARLES), Frédérique Biennier (INSA), Nicolas Salatgé (EBM), Bao Le Duc (FT), Antonin.Chazalet (FT), Pierre Chatel (TCF/LIP6), Tran Huynh (TCF), Alain Ozanne (INT)	
Final version	Nikolaos Georgantas (INRIA ARLES)	V1.0, 27/09/2009

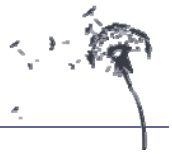


Table of Contents

1. Introduction

This deliverable, despite its title focusing on the security aspect, provides a pretty comprehensive presentation of the whole service composition approach within SemEUsE, dealing with the following aspects – steps in service composition:

- Description of the abstract user task (i.e., task requested by the user) and of the services available to be used for the fulfilment of this task. This description concerns both the functional and non-functional properties, including security. Comparative non-functional description between what is *required* and what is *provided* takes the form of an SLA.
- Service discovery for composing the concrete executable user task based on both the functional and non-functional above descriptions.
- Service composition that, more specifically, assures the global QoS requirements for the executable user task.
- Generation of the executable user task expressed in BPEL, where special attention is paid on integrating customized annotations and BPEL *Extension Activities* concerning security and enabling Late Binding and runtime Monitoring of services.
- Deployment of the executable user task, which includes SLA and Monitoring deployment.
- User task execution, including Late Binding of services, service data I/O on-the-fly adaptation, and late security binding.

The above outlined approach is illustrated in the following message sequence diagrams.

The work presented in this deliverable is complemented by appropriate references to other SemEUse deliverables. Further, perspectives and work still to be done in the rest of the project duration are identified.

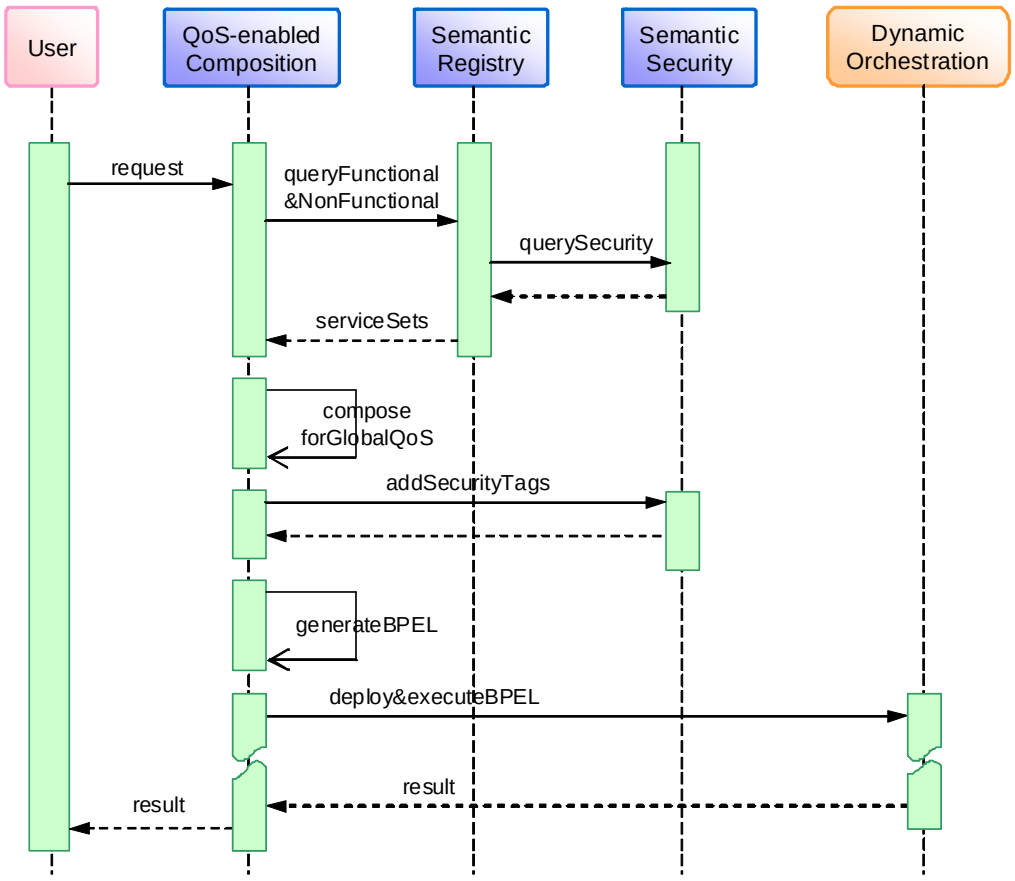


Figure 1-1: Service discovery and composition, user task generation

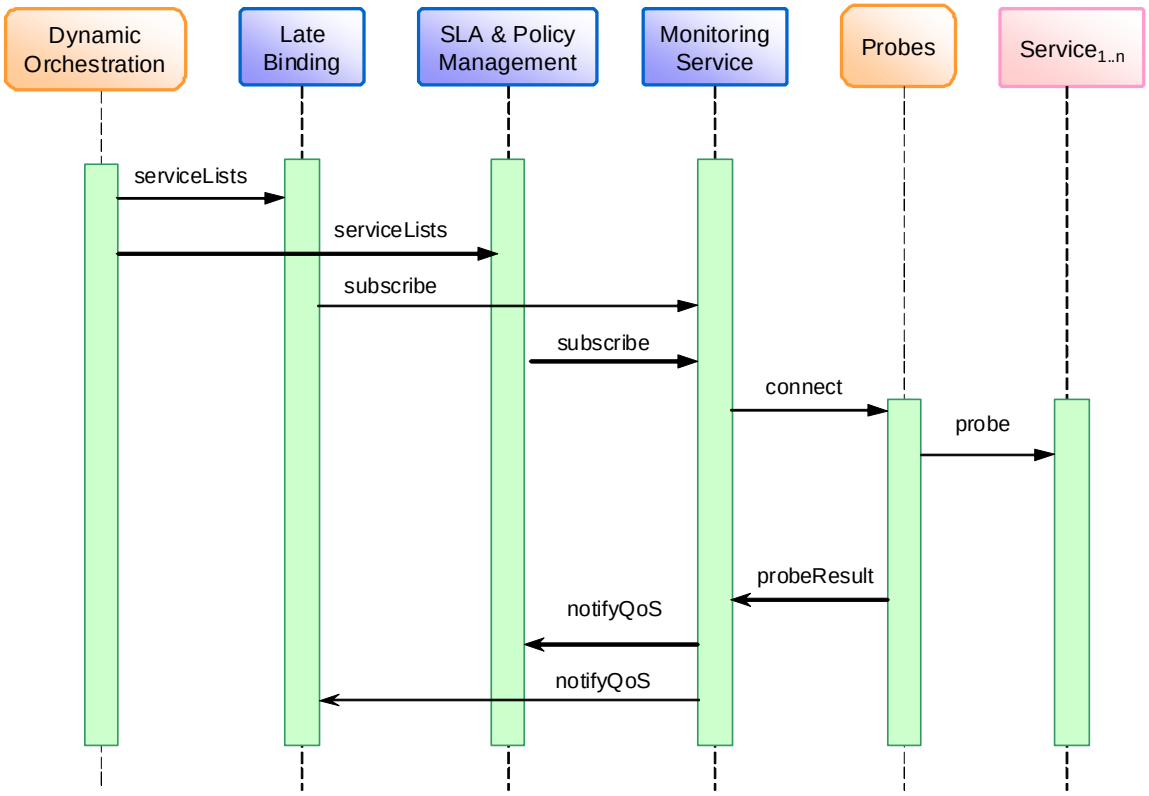


Figure 1-2: User task and monitoring deployment; monitoring

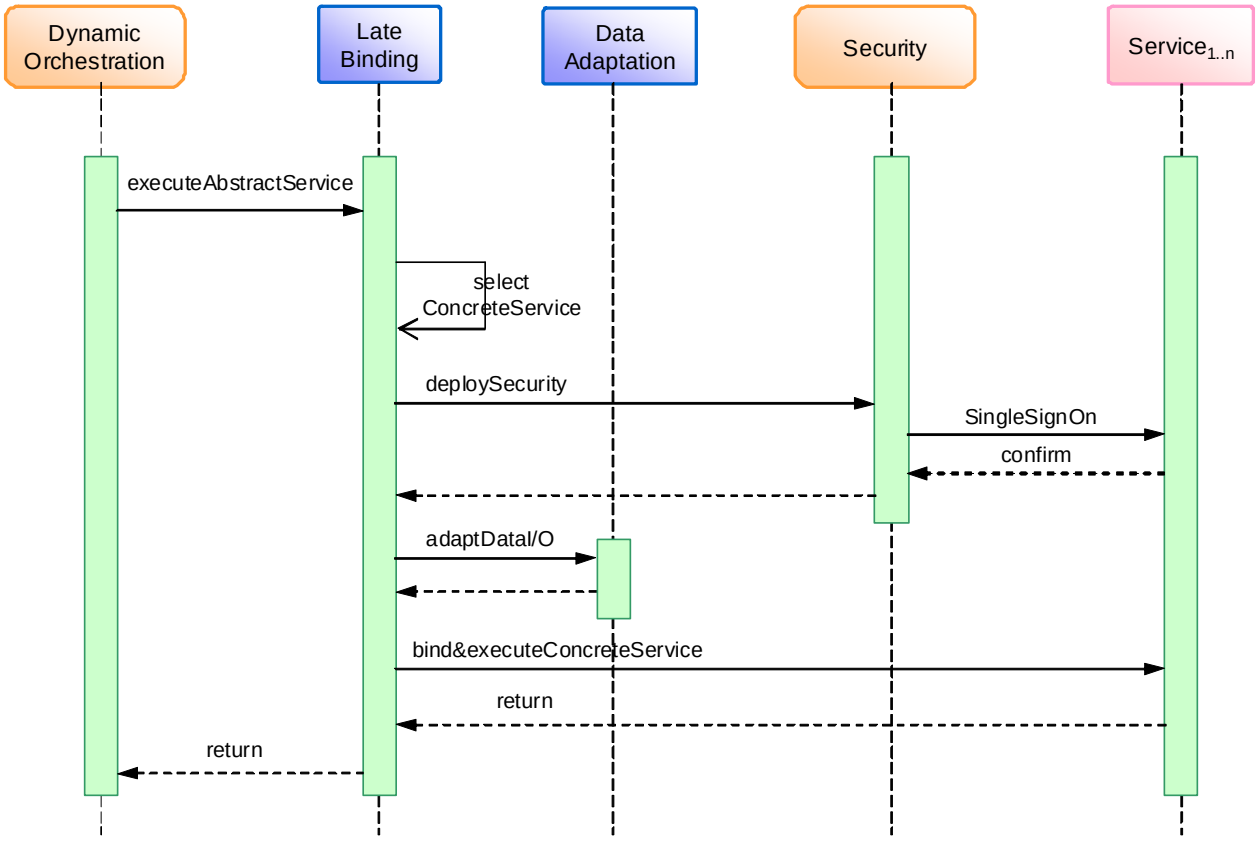
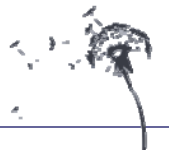


Figure 1-3: User task execution



2. User task and service description

2.1. Functional description

2.1.1. User task description

Our composition approach in SemEUsE is mainly based on the BPEL specification. The usage of BPEL is motivated by the fact that it allows defining service compositions with different abstraction levels. Previous studies focusing on modelling service compositions [16] divides composition languages into three classes: (i) Abstract level languages, (ii) concrete level languages and (iii) languages covering both levels, namely BPEL which represents according to the authors, a typical service composition language.

BPEL provides a sub-specification named Abstract Process which allows for defining processes with high level information. In our context, we use Abstract Process to define the user abstract tasks. The importance of this approach is threefold. First, it is based on standard specifications for service composition. Second, it enables further transformation of abstract tasks into concrete (i.e., executable) BPEL processes by binding abstract functionalities required by users to concrete services. Third, Abstract Process can serve as a very precise formal documentation for human understanding.

Abstract Process uses the notion of opacity to omit details of service compositions that are not relevant at the user level. According to this, BPEL elements (e.g., activities, expressions and attributes) can be replaced with opaque tokens (i.e., ##opaque). Nevertheless, Abstract Process is a general specification that covers several usage contexts and the notion of opacity can express different abstraction degrees. Therefore, the usage of the Abstract Process specification is delimited by the concept of ***profile***. Indeed, Abstract Process defines different profiles specifying the rules that determine the usage of the opaque tokens.

For the purpose of this work, we use the ***Template profile*** which allows for describing high level representation of compositions and hides their execution details. The Template profile allows the partners, the activities and the attributes used for identification of variables and message constructs representing partner interactions to be opaque.

The following example illustrates the usage of Abstract Process (i.e., with respect to the Template profile) to specify some elements of the fire fighting use case of SemEUsE.

```
<process name="FireFightingEmergencyProcess"
xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/abstract"
targetNamespace="http://www-rocq.inria.fr/arles/semeuse/bpel/examples"
xmlns:tns="http:// http://www-rocq.inria.fr/arles/ semeuse/bpel/examples "
suppressJoinFailure="yes"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ext="http://example.com/bpel/some/extension"
xmlns:template="http://docs.oasis-open.org/wsbpel/2.0/process/abstract/simple-template/2006/08"
abstractProcessProfile="http://docs.oasis-open.org/wsbpel/2.0/process/abstract/simple-template/2006/08">

<extensions>
<extension namespace="http://docs.oasis-open.org/wsbpel/2.0/process/abstract/simple-template/2006/08"
mustUnderstand="yes" />
</extensions>
<partnerLinks>
<partnerLink name="FireFighters" partnerLinkType="##opaque" myRole="CrisisManager"
partnerRole="FireFighter">
<documentation> The 'CrisisManager' and 'FireFighter' roles can be replaced by opaque tokens if they cannot
be specified </documentation>
...
</partnerLink>
</partnerLinks>
<variables>
<variable name="FireFightingRequestVar" element="##opaque" />
...
</variables>
```

```
<sequence>

<opaqueActivity template:createInstance="yes">
<documentation> This opaqueActivity gets fire information from different sources. It can be a receive activity for instance.
</documentation>
</opaqueActivity>
<assign>
<documentation> Transform fire information into the format accepted by the fire fighters' service.
</documentation>
<from opaque="yes" />
<to variable=" FireFightingRequestVar"/>
</assign>

<scope>
<faultHandlers>
<invoke partnerLink="FireFighters" operation="extinguishFire" inputVariable=" FireFightingRequestVar"
 ext:uniqueUserFriendlyName="invoking the extinguish fire operation of the fire fighting service"/>
...
<catchAll> <exit /> </catchAll>
</faultHandlers>
</scope>
...
</sequence>
</process>
```

2.1.2. Service description

For the description of the service, the reader can refer to Chapter 4 of the deliverable T2.1D1 "Functional properties, user preferences and context information semantic description".

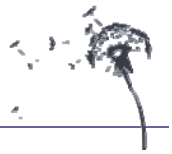
2.2. Non-functional description

2.2.1. Overall QoS model

QoS in Dynamic Service Oriented Environments

In Service Oriented Computing (SOC), providing services with quality guarantee capabilities requires having solutions that include Quality of Service (QoS) models as a key feature. Indeed, QoS models provide the appropriate ground for QoS provision in service oriented environments. In such environments, QoS can be influenced by several factors including the hardware and network infrastructure (capabilities of computing devices, network connectivity), the quality level provided by application services and the end-user characteristics (mobility, generated traffic). This implies that, in order to obtain an accurate evaluation of QoS, none of these aspects should in principle be neglected at the QoS modeling stage. Thus, QoS should be considered on an end-to-end basis, meaning that QoS models should cover quality features of all the resources and actors involved in fulfilling a service. In practice, these include (i) the service environment and its underlying hardware and network infrastructure, (ii) application services and (iii) end-users. Furthermore, QoS models should consider emerging features related to the dynamics of service environments. Indeed the large success of mobile devices and wireless networks has made service environments more dynamic, thus QoS models must change accordingly. For instance, emerging features such as the adaptability and context awareness of application services should be considered as QoS properties affecting the quality level required by users.

A third issue concerns the expressivity of QoS models. Related to this, a promising approach addressing QoS modeling relies on the Semantic Web technologies, notably ontologies. Ontologies allow for capturing and defining QoS knowledge with rich semantic information. The importance of such representation is two-fold: First, it is a machine-understandable specification for QoS that provides the appropriate ground for several service engineering capabilities such as logical reasoning on QoS and automation of QoS management in dynamic service environments. Second, as they represent a common and shared knowledge of QoS, ontologies allow for broadening QoS understanding across different specifications that address QoS for varied purposes, thus enabling to cope with the syntactic heterogeneity of QoS specifications.



Despite the considerable amount of research devoted to QoS ontologies, none of them considers jointly QoS on an end-to-end basis and emerging QoS features related to the dynamics of service environments. On one hand, we have ontologies tailored to a specific aspect of QoS centered on the service level [8,6,2]. These ontologies do not consider other important resources involved in services' provision such as the network and hardware infrastructure. On the other hand, we have 'general' QoS ontology languages [13,3] that specify basic concepts for QoS and allow defining additional concepts. These languages are not representative QoS models since they do not give any tangible specification of QoS. To the best of our knowledge, there are no QoS ontologies supporting the dynamics of service environments.

This section introduces a semantic QoS model that copes with the above issues, i.e., it considers QoS on end-to-end basis and puts a special emphasis on emerging QoS features related to the dynamic nature of service environments. Our model comprehends four ontologies that specify: (1) General and basic concepts needed for QoS description, (2) QoS properties related to the service environment and its underlying hardware and network infrastructure, (3) QoS properties of application services, and (4) QoS at the end-user level.

For ontologies (1) and (3), we recall QoS properties defined by the Web Service Quality Model (WSQM) [10] which is a prominent standardization effort proposed by the OASIS WSQM technical committee for the specification of Web Services' QoS. It defines a well-founded taxonomy of QoS and provides a wide range of QoS properties. WSQM employs a number of terms that have specific meanings to QoS. In this paper, we adopt this terminology and we list it below with explicit definitions. We use the term quality factor to refer to an attribute used to represent and assess QoS, the term quality group to refer to a group of quality factors concerning a common aspect of QoS, and the term quality item to refer to an atomic attribute within a quality factor (e.g., the response time quality item fall under the performance quality factor, the confidentiality quality item fall under the security quality factor).

WSQM Overview

WSQM (Web Service Quality Model) [10] is a conceptual model for Web Services quality that defines three basic concepts: quality associates, quality activities and quality factors. A quality associate refers to the person or the organization that is directly or indirectly managing QoS. The set of actions performed by quality associates throughout the whole services' lifecycle are called quality activities, whereas the quality factors are the attributes used to represent and assess QoS.

Our primary focus is on the quality factors which represent the core entities of QoS. Indeed, QoS models are usually centered on the definition and the classification of quality factors. WSQM divides quality factors into three quality groups with respect to system architecture, operation and business perspective:

- *System Information* is a set of static information on systemic functions that are defined and evaluated before using the service. It includes the Security, Suitability for Standards, Manageability and Business Process quality factors.
- *Service Measurement* refers to quality factors measured while a consumer is using the service. It is mainly about the Performance and Stability quality factors.
- *Business Value* assesses the service from a business point of view. It includes the Service Cost, Service Suitability, Service Aftereffect and Service Brand Value quality factors.

WSQM formally specifies the quality factors and their associated concepts using the Web Service Quality Description Language (WSQDL). WSQDL provides a detailed QoS specification, however it represents some shortcomings with respect to dynamic service oriented computing. Next, we give the details of WSQDL then we assess it regarding our goals.

WSQDL Overview

WSQDL provides a XML-based description method for standardizing the expression of QoS exchanged between application services and their consumers. It defines the constructs needed to specify the quality factors, their taxonomy, the way they are assessed and the relationships between them.

The main construct within WSQDL is QualityFactor. It is represented with a global structure that expands its inherent quality items into four levels reflecting the complexity of quality factors and allowing for the specification of fine granularity details of QoS. These levels are: *SubFactor*, *Property, SubProperty* and

Function. For instance, the Security quality factor is divided into three properties: confidentiality, integrity and non-repudiation. Confidentiality may include other sub-properties such as message level confidentiality and user confidentiality. By turns, the message level confidentiality can be defined using the XML-encryption function.

In addition, QualityFactor divides into four disjoint subclasses with respect to the way it is assessed: *MeasurementFactor, EvaluationFactor, BusinessValueFactor* and *BusinessProcessFactor*.

MeasurementFactor (e.g., Performance) defines quality factors that can be quantitatively measured using metrics. The metrics are defined with the *MetricType* construct which is given in terms of *MeasurementFunction*, *MeasureDirection*, *EnvironmentVariables* and *Metric*. The Metric construct is defined over the *MetricValue* and *Unit* constructs.

EvaluationFactor describes the type of appraisable quality factors determined by their conformity to an appraisal standard. The main construct defining this factor is *Conformity* which describes the degree of conformance between a quality factor and a standard specification. The Security, Manageability and Interoperability fall under this type of factors.

BusinessValueFactor addresses quality factors used to determine business aspects of QoS. In contrast to EvaluationFactor, it has no standard evaluation and it may be assessed by users who estimate the business value of a quality level. Therefore, BusinessValueFactor is defined using the *UserAppraisal* construct.

Finally, BusinessProcessFactor defines the set of quality factors assessed by their ability to achieve business goals.

Assessing WSQM for dynamic service computing

Although it provides a well-founded specification of QoS at the service level, WSQM presents three main shortcomings regarding our goals: First, it neglects QoS associated to other resources and actors participating in service provisioning (e.g., network, devices and end-users). Second, it does not consider emerging QoS factors related to the dynamics of service environments (e.g., adaptability, context-awareness). Third, as already explained WSQM adopts an XML-based approach to specify QoS, which makes it subject to the syntactic QoS specification problem.

Regarding the above issues, our approach aims to define a semantic QoS model that addresses all elements of service environments and takes into account the dynamic nature of these elements. Seeing the importance of WSQM, our model makes use of this standard specification to define QoS at the service level.

The proposed model is defined over a set of ontologies underpinned by the Web Ontology Language (OWL), which is a W3C recommendation designed for publishing and sharing ontologies. The next section details the developed ontologies and outlines their usage in the context of dynamic SOC.

A Semantic End-to-End QoS Model for Dynamic Service Environments

We present a semantic QoS model that addresses QoS on an end-to-end basis by covering quality factors of the main elements acting in dynamic service environments, in particular it deals with: (i) network and hardware resources, (ii) application services and (iii) end-users. Our model puts a special emphasis on QoS features related to the dynamic nature of these resources such as end-user mobility and adaptability and context awareness of application services.

Our model is designed according to a layered approach integrating certain modularity and flexibility requisites, thus aiming to provide distinct and easily manageable ontologies.

As depicted in Fig. 2-1, it comprehends four ontologies:

1. The QoS Core ontology incorporates general constructs needed for QoS description (e.g., quality group and quality factor). The conceptual elements of this ontology are inferred from WSQDL.
2. The Infrastructure QoS ontology specifies quality factors related to the environment and its underlying network and hardware infrastructure. It focuses on the capabilities of mobile devices,

the connectivity of wireless networks and the characteristics of the environment where users and services behave.

3. The Service QoS ontology specifies quality factors of application services. It extends the factors already defined by WSQM with other factors (e.g., adaptation and context awareness) supporting the dynamics of the application services. The added factors are carefully selected by examining the dynamic nature of service environments. Additionally, this ontology is extensible in that new quality factors (e.g., domain-specific quality factors) can be easily added.
4. The User QoS ontology addresses user concerns about QoS. The role of this ontology is two-fold: First, it provides the constructs needed to specify user QoS requirements. Second, it specifies quality factors associated to the user such as user mobility.

The ontologies 2), 3) and 4) specialize the general concepts defined in the QoS Core ontology; they are layered (Fig. 2-1) from lower level (i.e., infrastructure) to higher level (i.e., end-user) aspects.

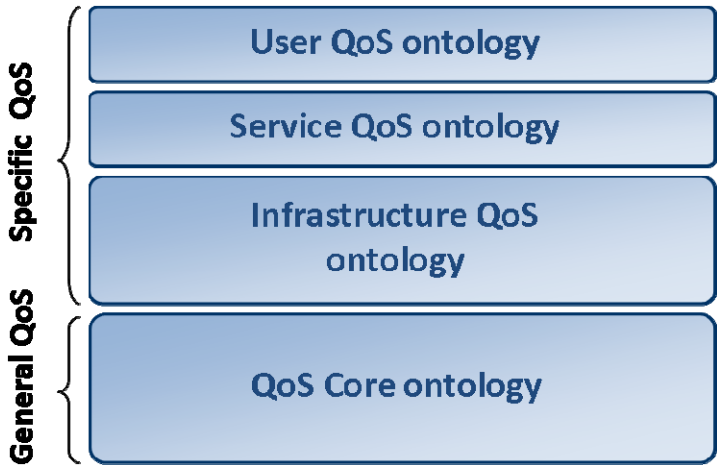


Figure 2-1. QoS Model Overview

QoS Core Ontology

In order to define base concepts required for QoS description, the QoS Core ontology makes use of the WSQDL standard schema, which, as discussed in Section 2.1, provides ample and detailed specification for quality factors, their taxonomy, the way they are assessed and the relationship between them.

The QoS Core ontology (Fig. 2-2) represents constructs from the schema with semantic information. The main constructs of this ontology are *QualityGroup, QualityFactor* and *QualityChain. QualityGroup* consists of one or more *QualityFactor*, which are represented from two key standpoints: their structure (i.e., SubFactor, Property, SubProperty, Function) and their type (i.e., MeasurementFactor, EvaluationFactor, BusinessProcessFactor, BusinessValueFactor). The *QualityChain* construct is defined over the *hasInfluenceOn* property indicating that two or more quality factors are in a quality chain relationship.

The QoS core ontology comprehends further constructs not represented in Fig. 2-2 such as MetricType, Conformity and their associated concepts. In this ontology, it is worth mentioning that the construct Unit within MetricType references other external ontologies to define its semantics. Indeed, the specification of measurement units is not addressed by WSQDL and it is not in the scope of the present work. Currently, Unit references the OWL time ontology [4].

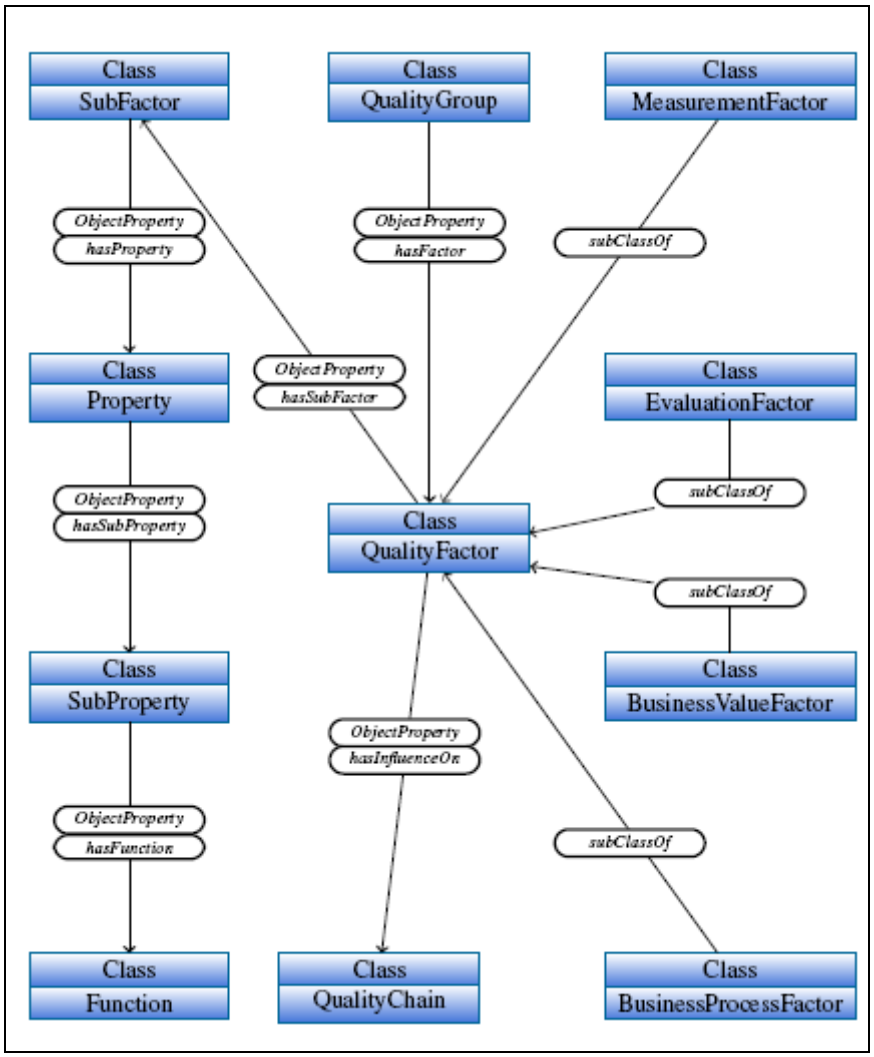


Figure 2-2. Overview the QoS Core Ontology

Infrastructure QoS ontology

As already explained, QoS is difficult to define and to evaluate in an accurate manner without considering the infrastructure supporting the provision of services, namely the network and the devices enabling services and users to interact, as well as to the characteristics of the environment where services and users act. For this matter, we developed the Infrastructure QoS ontology (Fig. 2-3) which consists of three quality groups: Network, Device and Environment.

The Network quality group addresses the communication infrastructure in dynamic service environments. It consists of two quality factors: Type and Performance. The first factor indicates the network topology, which can be either Wired or Wireless. The second factor comprehends quality items specifying the performance of a network. Common performance metrics include Bandwidth, Latency, Loss and Jitter [7]. Bandwidth refers to the rate of data transfer; Latency refers to the total time taken to deliver a message; Loss represents the rate of message units lost during the delivery of a message. Finally, Jitter expresses the variation in Latency.

The Device quality group addresses hardware devices hosting application services (e.g., Server) or supporting end-users (e.g., PDA, Smartphone, PC). It defines three quality factors outlining the capacity of devices: Category, Mobility and Performance. The first factor refers to the role of devices in service provision, which divides into two types: Client and Server [13]. The second factor describes the mobility of devices which can be either Stationary or Mobile. The third factor comprehends quality items specifying common device capabilities, i.e., CPU, Memory, Storage Capacity and Power Consumption.

The Environment quality group specifies quality factors intrinsically related to the environment where users and services exist. Such environments are populated with networked services supporting a wide variety of user applications. The quality of these environments can be assessed by evaluating their degree of support to user applications [5]. This can be specified using three quality factors: Service Density, Sustainability and Scalability [5]. Service Density refers to the number of services available in service environments. It indicates in part the ability of an environment to satisfy user requests, i.e., the higher the service density is, the higher is the probability to fulfill the users' tasks. Sustainability measures the environment's ability to sustain employed services if they fail. When a service part of the environment fails due to issues such as power limitations or mobility, the environment may set off some fault tolerance mechanism like identifying an alternative service which matches the original functionality. Scalability refers to the ability of the environment to support a large number of active users and to handle their requests in a satisfying manner.

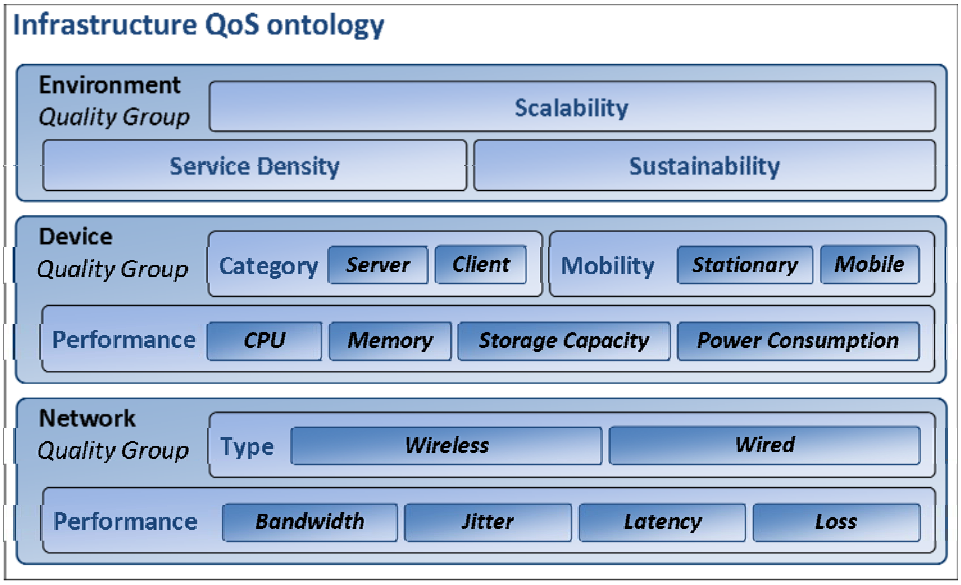


Figure 2-3. Overview of the Infrastructure QoS Ontology

Service QoS ontology

The Service QoS ontology is of fundamental importance in our model, since, in practice, the main QoS features are determined by application services. The key idea underlying this ontology is: (i) to recall WSQM standard quality factors for defining common QoS features of application services and (ii) to specify other quality factors to address the dynamic nature of services.

As already mentioned, WSQM defines three quality groups (i.e., System Information, Service Measurement and Business Value). In addition to these groups, we define a quality group called Dynamic Capabilities which defines quality factors supporting the dynamics of service environments. The responsibility of application services with respect to the dynamics of service environments includes supporting adaptation, context awareness and facilitating the automation of user request processing.

- Adaptation: Services operating in highly dynamic environments need to continuously adapt themselves in order to react to changing conditions such as environmental conditions and user requirements [11]. To this regard, we define the Adaptation quality factor which addresses the ability of a service to adapt itself to changing conditions and to reconfigure itself accordingly.
- Context-awareness: Dynamic service computing focuses on fulfilling user tasks in a transparent way that places few demands on user attention. Invisibility of services can be accomplished in part by reducing input from users and replacing it with knowledge of context. Context awareness is further required to enable adaptation to changing conditions by exploiting context information, such as location and proximity to other devices and services. To address context awareness of application services, we define the Context quality factor, which describes the ability of a service to gather, manage, use and disseminate context information.
- Automation Support: Dynamic service environments focuses on fulfilling user tasks on-the-fly (i.e., at runtime) by dynamically locating and integrating available services. Hence, services' discovery,

selection and composition in such environments need to be performed automatically, i.e., with little or no human intervention. Such assumption requires in part the description of services' semantics using a machine-understandable specification. To this regard, we define a quality factor named Automation Support, which describes the ability of a service to support automated management. It consists of a quality item named Semantic Information Offerability denoting the ability of a service to provide semantic description of its functional and non-functional features.

Besides the Dynamic Capabilities quality group, we define the Domain-Specific quality group to address quality factors associated to services' domains. Indeed, service oriented computing deals with open environments offering access to a wide variety of services from different domains. Further, users often need to address QoS factors related to particular domains. For instance, if we consider a multimedia application which diffuses entertainment videos, it will be pretty useful to users to know the "encoding quality" of the given videos in order to choose the appropriate media player or to select the right screen resolution. The Domain-specific quality group addresses such issues by allowing the definition of any domain-specific quality factor as long as the latter references concepts within appropriate domain ontologies to specify its semantics.

The main conceptual elements of the Service QoS ontology are depicted in Fig. 2-4. In this figure, the layering goes from lower level aspects (i.e., system) to higher level aspects (i.e., domain-specific). The white boxes within the figure represent the quality factors already defined by WSQM, whereas the colored boxes represent the new factors defined for dynamic and domain-specific application services.

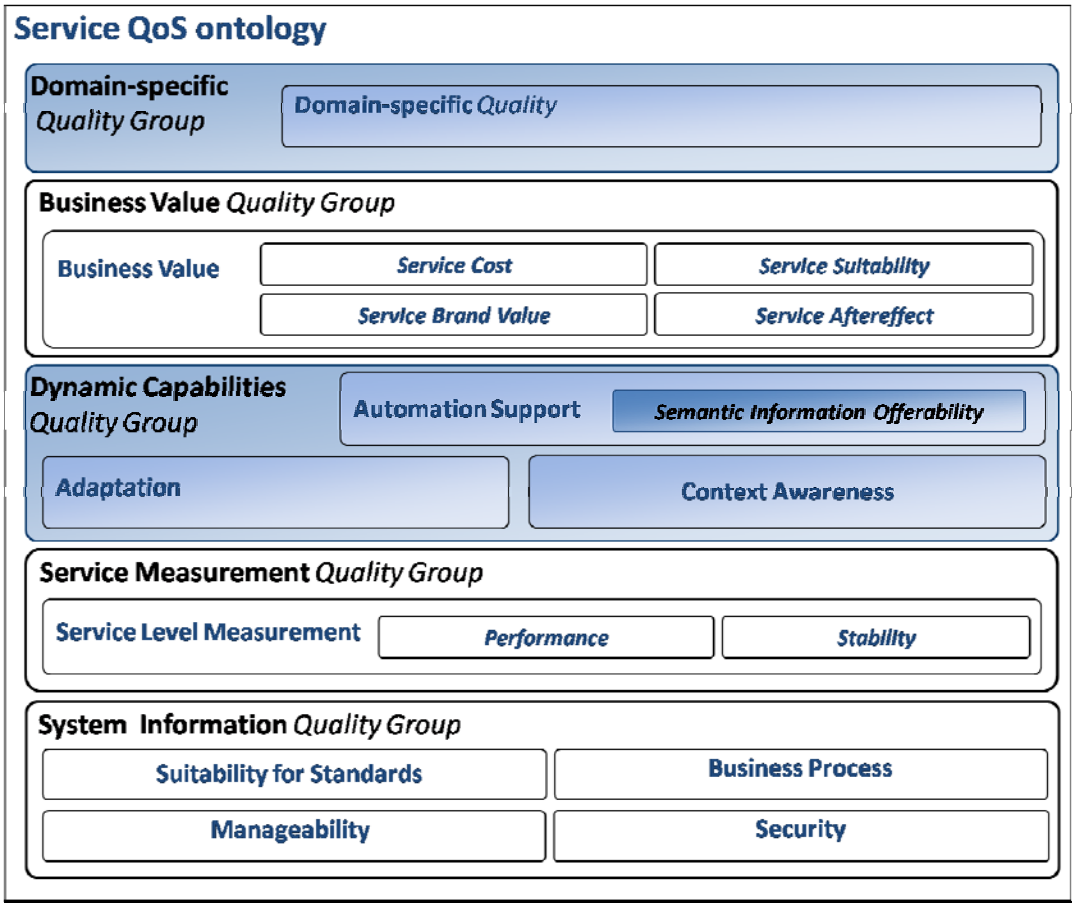


Figure 2-4. Overview of the Service QoS Ontology

User QoS ontology

This ontology addresses users' concerns about QoS. It is composed of two main components (Fig. 2-5): (i) user requirements modeling and (ii) user profile modeling. Requirements modeling allows users to express their QoS requirements according to their own quality experience [15]. These requirements are formulated as a group of constraints defining the QoS level required by users. At the heart of

requirements modeling, we distinguish the construct Requirement, which corresponds to the description of a constraint made by the user on the offered QoS. The constraint targets a QualityFactor from the QoS Core ontology, and it is given in terms of Operator, Value and Unit. The latter constructs change according to the type of quality factors.

Requirements dealing with EvaluationFactor, BusinessValueFactor and BusinessProcessFactor are expressed using boolean operators (i.e., is-a, is-not-a) and string values, whereas requirements addressing MeasurementFactor are given in terms of comparison operators (i.e., equal, not-equal, more-than, less-than, max-value-of, min-value-of), numerical values and measurement units. Users can further define composite requirements using *and* and *or* operators. Further, they are allowed to express their relative preferences about QoS requirements by assigning a certain Weight to every requirement.

Let us now consider the user profile modeling which aims at defining user-related quality factors (e.g., user mobility, user generated traffic). User Profile comprehends two quality factors: Mobility and Traffic. These factors have been defined based on the model proposed by Resta et al. [14]. User Mobility is divided into three patterns: Stationary users, QoS-driven users and Mobile users. The first pattern concerns really stationary users or users with constrained movements that does not affect service provisioning. QoS-driven users are mostly stationary but they move when their perceived QoS level drops below an acceptable threshold. Finally, the mobile users are characterized by continuously moving positions.

Concerning User Traffic, users are divided into three different classes of load according to their generated traffic: low, medium, and high load. The lowest class of traffic accounts for users who are using the network for e-mailing, chatting, and light Web browsing. The medium class of traffic accounts for users who are using the network for intensive Web browsing, file downloading, audio streaming, and so on. Finally, the highest class of traffic accounts for users who make an intensive use of the network, such as video streaming.

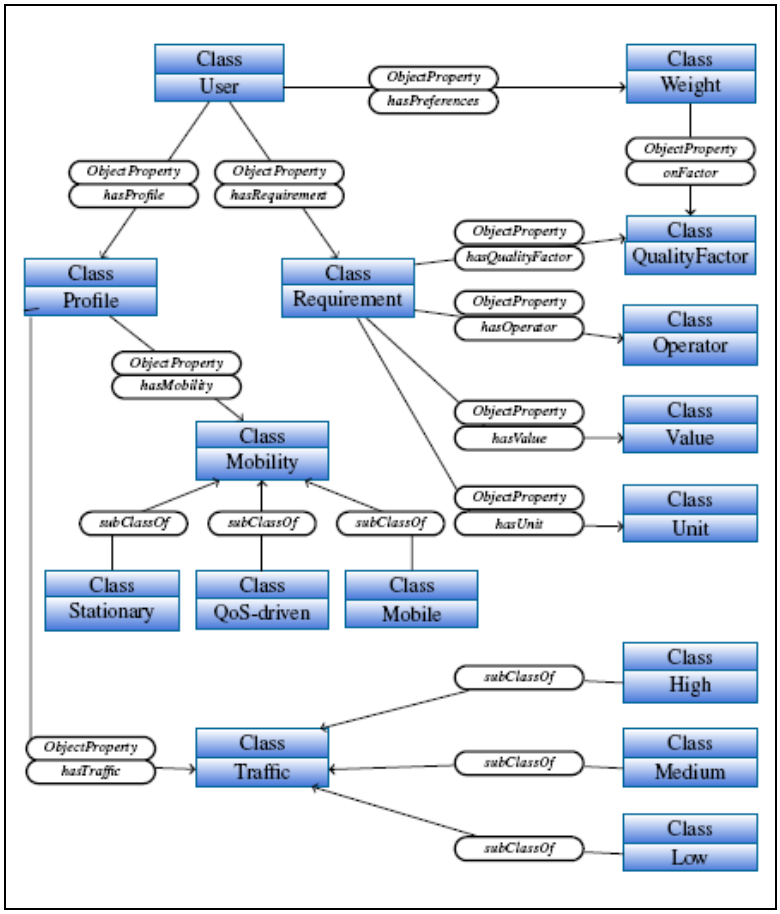


Figure 2-5. Overview of the User QoS Ontology

Usage of our model

Our model concentrates on QoS knowledge representation rather than a language to specify QoS. To this extent, our approach has been to decouple QoS knowledge from QoS specification by providing separate and reusable ontologies. Thus any appropriate QoS specification language can be used on top of our QoS model. More specifically, to employ our model for QoS description, the introduced ontologies have to be referenced by syntactic QoS languages that additionally support semantic annotations, notably XML-based languages enriched with OWL semantic annotations [12].

Referencing QoS semantic concepts allows for coping with the syntactic heterogeneity problem and yields semantically enriched QoS descriptions that combine the accuracy of syntactic languages with the rich semantics of QoS ontologies.

This approach is important in broadening QoS understanding between heterogeneous specifications addressing QoS for different purposes. Thanks to our model, these specifications can reference common ontological concept and share the same vision of QoS. As depicted in Fig. 2-6, our model can be used to express user QoS requirements and to specify the QoS level provided by services.

The latter point leads to another, increasingly important, aspect, which is enabling service providers and users to agree on a contract concerning the quality of the offered service (i.e., Service Level Agreements SLA). Specifically, our model is useful for both the contract specification and contract enactment. In the first case, our ontologies can be used to define QoS terms of an SLA, or other terms that define aspects such as user device capabilities. Once the contract is defined, our ontologies can be also used to monitor SLAs and to adapt them to changing conditions in a dynamic way. The need for dynamic SLA specification, negotiation and monitoring is getting increasingly important in SOC [15]. The proposed model is therefore a general framework (Fig. 2-6) that provides the appropriate ground for several engineering capabilities including QoS requirements engineering and QoS-based service engineering (e.g., service discovery, SLA management, monitoring).

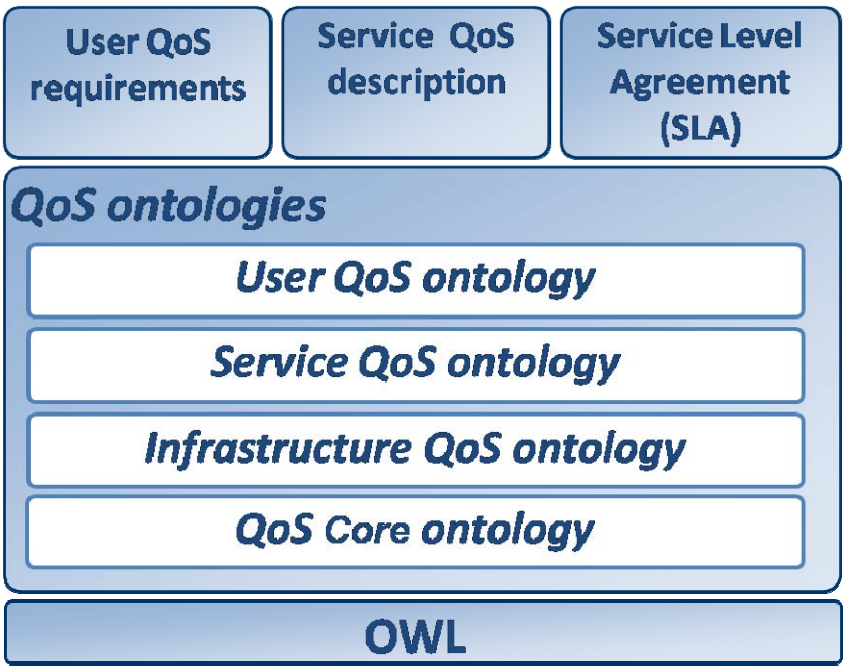


Figure 2-6. QoS Specification Framework

Our model is extensible in that domain-specific QoS factors can be easily added. Furthermore, the proposed model focuses on representing QoS knowledge with rich semantic information rather than specifying a language for QoS. Coupled to XML-based QoS specifications, the developed ontologies can formulate a robust QoS description framework that combines the rich semantics of QoS ontologies with the accuracy of QoS specification languages.

2.2.2. User task, service and SLA description

SLA enriches a service definition as described in the WSDL. While a service description, for example, using WSDL defines the service interface relationship between a service and its using application, the SLA defines the agreed QoS characteristics and the way to evaluate and measure them. SLA complements service description languages. Service descriptions are input to the design and implementation of the service system and the client application using its service.

WS-Agreement's [66] service contracts contain guarantee scopes, objectives constraints and business values such as penalties and rewards that can be used for enforcing contracts' objectives

In the context of the SemEUsE project, the syntactic integration of QoS contracts and its binding with semantic is based on the WS-agreements standard [66] defining contracts attached to WSDL descriptions of services APIs. This approach is explained in Deliverable T2.2D1.

We use WS-Agreement for describing the QoS both for the services (provided) and the user task (required). This approach is presented in Deliverable T2.2D1. We have extended WS-Agreement semantically by linking to our QoS model presented in the previous section.

2.2.3. Security model

Due to the high patrimonial value of the enterprise information system, a security strategy must be set up to protect the "intangible" part of the information system. This leads to design a secured information system taking into account the particular assets and risks so that vulnerabilities can be defined. In cross- organisation context different information systems having different Quality of Protection (QoP) preferences and security policies interoperate to carry out collaborative services.

Security objectives are traditionally described according to the 3 basic security services, namely confidentiality, integrity and availability. These basic features are extended to integrate authentication, non repudiation and access control mechanisms. As far as the service environment architecture is concerned, the OASIS Service Reference Model proposes a multi-layered security architecture:

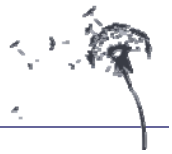
- Network security refers to the infrastructure organisation. This part mostly addresses Deny of Service (DoS) attacks, leading to non-availability. This part makes a heavy use of QoS monitoring and on SLA control so that DoS attacks can be brought to light.
- Transport security: this part refers to secured communication channels between ESB nodes and between user and ESB. It is mostly focused on confidentiality (using secured and crypted protocols) and integrity (due to message signature). We can also note that message signature fulfils also the non repudiation requirement
- Application security focuses also on the confidentiality requirement, taking into account access rights to allow (or not) service invocation and protected data access.

In order to provide a end-to-end security model,

State of the art

Approaches and Methods to set up Security Strategies

Securing corporate information system requires identifying threats and vulnerabilities, defining a convenient access control policy by imposing constraints on resource access according to business strategies depending on requestors' identity (i.e. identity claims, roles, preferences, context…etc.). The definition of a global and consistent security strategy leads to deploy security policies to control access to resources based on actors' roles within the enterprise organization structure (i.e. organizational roles). These roles can then be integrated into enterprise workflow specifications and user authentication systems. Consequently, security requirements must be taken into account simultaneously while designing both the business processes and the physical implementation. As result, building a security policy is often reduced to the implementation of a "secure infrastructure". For this purpose methods and standards have been defined since the 80s (i.e. the DoD Rainbow series [47], International level of certification [49], Common Criteria [45]) and unfortunately they only provide technical solutions to technical identified threats and technical vulnerabilities whereas organizational vulnerability must be also taken also into account to define consistent security requirements as proposed in the ISO 17799 standard [51]. In response to perceived risks, adapted policies can be set up using different methods such as EBIOS [46], MEHARI [43], OCTAVE [36] and so on. On the other hand, well-defined risk analysis and assessment methods seem to be helpful to design security requirements in order to build survivable consistent and reliable systems as proposed in SNA [55]. Nevertheless these methods are designed for bounded or isolated information systems and prove to be inadequate to fit collaborative business



environments and deal with interconnected partners' information systems. Due to the openness and agility requirements involved by the collaborative business context, contextual security strategies should be defined and embedded in information system components in such a way the inter-enterprise information flows can be secured. The interoperability requirement between various different partners' information systems is often fulfilled by setting up service-oriented architecture integrating adapted security components.

Standards and Security Components in Service Computing

As services are associated to information system assets, each service or resource must include its capabilities and requirements by deduction from a global security strategy and transforming them to security policies (i.e. a set of assertions on used resources and/or activity enactment) which are mostly focusing on access controls.

Due to the intrinsic flexibility and openness of services environment, each service or resource must adapt the security policy to the execution context, the confidence level between the source and target systems must be also taken into account. To achieve this goal, the Reference Architecture defined by the OASIS [58] proposes a trust model between services stakeholders and service customers (i.e. participants) in order to define who can perform an action or trigger an event. This leads to define several trust level associated with different "social groups" mixing stakeholders and participants and relationships between these domains. In addition, these trust levels are used to define access policies in terms of authorization and authentication strategies so that credentials (integrating roles) and identity (including identifier management) can be used. Trust relationships between participants can also be used to deduce decentralized authentication and authorization mechanisms so that trust chains can be established within a global service chain.

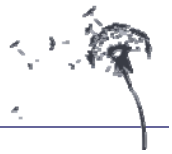
As a consequence, these security choices impact the implementation of security measures. Applying security basic services in an end-to-end communication involves the protection of exchanged information, services and business processes deployed on servers. Current solutions to express security preferences and requirements tend to be technical low-level tendency and not user-friendly which lead to express requirements into different topics as proposed in [58]:

- *Transport security which* relies on secured communication infrastructure elements or secured communication protocols
- *Message security* which mostly relies on authentication and authorization tokens to set up identity-based access control
- *Data security* which includes onsite data encryption and digital signature management
- *Environmental security* which is mostly devoted to the "process" organization.

These service-level protection topics require dedicated standard and appropriate tools to implement the basic security services. For example, XACML [57] allows the description of access control specifications, SAML [56] is used for exchanging authentication and authorization data between security domains whereas WS-Trust and WS-Federation provide solution to implement single-sign on facilities along a service chain. Nevertheless, these approaches cope with setting up interoperable security protocols and components rather than the warranty of an end-to-end protection that respect the global security strategy.

Role-based access control

The definition of a security strategy leads to deploy security policies to control access based on actors' positions within the enterprise organisation structure i.e. their organisational roles, these roles can be integrated into enterprise workflow specifications and user authentication systems [54, 53]. Role-Based Access Control (RBAC) [50] is a security model which relies on the enterprise organization view to provide authorization to access resources. The RBAC model creates an indirect relationship between rights and actors through the roles played by the actors. Thus, security policies are defined to govern roles, and facilitate the management of security policies. RBAC affects users of the roles they can play; these roles are associated with permissions which provide access to resources [40]. This model allows establishing high level organizational policies by adding constraints on the top of the relations between its components. It allows also for each principal to get multiple roles, and then choose to activate one or more of the assigned roles during each session. However, the RBAC model does not take in consideration the notion of execution context. To do so, the work in [61] extends the RBAC for adaptive process management systems by defining new type of access rights –in addition to user dependent access rights– called in this work process-dependent access rights, this allows changing permission according to the current executed process. The DRBAC [65] proposes dynamic roles to access resources by defining policy context sets which allow triggering changes in role state via role state machines according to context changes. The Generalized Role-Based Access Control (GRBAC) model [44] extends the role notion to be applied to all system entities i.e. subject roles, environment roles, and object roles. Contextual information is checked to grant access according to entities roles. However, both DRBAC and



GRBAC do not consider the collaborative business context where dynamic ad-hoc groups of users of different policies and contexts conjointly work to carry out a common objective. Additionally, due to the multitude of system entities, management of roles in GRBAC becomes difficult to maintain. The work in [59] extends the RBAC model with the physical location of resources and actors to decide whether a role has an access right to resources. [64] proposes role-based access control security architecture to support distributed and multi-organisational enterprise. Though this work presents well-designed and integrated policy-based security architecture, it suffers from some drawbacks: the notion of context has been neglected which, in this case, impacts the totality of the architecture. Additionally, the work treats the problem from a purely security-oriented viewpoint without taking into account the business dimension such as process or workflow and their related problem of coordination and dynamic configuration. In this order, the RBAC model is extended to the W-RBAC [60] by adding the "doer" relationship as well as various dynamics to monitor the integrity of workflow. Lastly, Roles can also be added in the BPEL task code so that authorisation can be verified [63]. Despite of the specification of several roles in the BPEL4RBAC architecture [62], this extended role notation does not take into account role semantic relationships. To overcome this limit, semantic annotations must be added.

Discussion

Setting an end-to-end security strategy adapted to the collaborative process context should take into account different partners' security strategies and evaluate the global consistency of the forecasted business process organization: each partner must fit the global protection requirement. This involves redesigning the security strategy according to the collaboration context and adapts different security methods accordingly.

At the service level in a service-oriented architecture, different protocols have been defined to support technological security requirements, but they lack their ability to integrate enterprise security policies which can be conveniently defined thanks to a systematic threat analysis (as proposed in the EBIOS, MEHARI and Octave methods). Nevertheless, these methods, except SNA, lack the integration of security requirements at the design-time of business processes, the propagation of these security and reliability constraints during the process orchestration in a distributed service chain. The federation strategy proposed in WS-federation only focuses on the technical security "realms" federation, which results in different security perimeters.

To overcome these limits, we propose to integrate security requirements derived from the global security and reliability policies and including service context and user preferences, so that security technical components can be orchestrated while building dynamically the convenient service chain and the associated protection level agreements between partners.

Designing and maintaining a consistent security policy and a safe information system requires first to define the enterprise exact needs, then to identify the threats and evaluate the "non security" costs and finally to design the adapted infrastructure. Different methods and tools (EBIOS, CERT/Octave, SNA, Safe/CISCO…) are already available to set a security strategy, but none of them are able to support a security project from the early design step to the technical implementation (Figure 2-7). Moreover, these methods are designed to define and implement a convenient security strategy for an enterprise, focusing on the information system infrastructure (mostly the network organization, integration of "secured" software components and SSO or RBAC systems). These different elements fit well "static" and intra-enterprise context but they do not fit the inter-enterprise collaborative context as BP are built dynamically depending on the context.

	Requirements analysis	**Design**	**Implementation**
EBIOS	Risk and objectives identification		
OCTAVE	Structured information access identification	Objectives prioritization Best practices	Audit and implementation project management
SNA	Process and resources workflow identification	"Survival process" design	
MEHARI	Shortened risk analysis	Best practices	Implementation project management

Figure 2-7: Methods and their contribution to a security project

Organisation of a security model

Designing and maintaining a consistent security policy and a safe information system requires first to define the enterprise exact needs, then to identify the threats and evaluate the "non security" costs and finally to design

the adapted infrastructure and organisation. To support these requirements, the SEMEUSE security model integrates a security requirement identification method, adapting existing methodological environment to the service world, the specification of these security requirements in a contractual environment (the Quality of Protection Agreement) to provide a legal basis as well as the development of semantic security annotation to improve the service description and consequently the discovery process leading to the security framework we propose in Deliverable T2.2D2.

Security requirements identification

To fulfil the multiple context requirements involved while designing inter-enterprises business processes, we add the following steps to a traditional security project organisation (from the vulnerabilities and risks evaluation, security objective identification, design and implementation steps for the security policy) conducted by assembling the different above mentioned methods:

- **Trusted community identification:** this part is devoted in identifying different trusted communities including potential partners. This step is based on a dedicated "partnership risk" evaluation including the following criteria: concurrent or complementary partnership, knowledge on the partner, importance of the collaborative process in the enterprise organization… according to the collaboration status [39]. In our emergency use case, different communities are associated to the different domains (medical, firemen, police, media…) so that legal and organisational security constraints can be taken into account globally.
- **Patrimonial evaluation:** This part is used to gather objective and subjective value of both information units and business processes to identify security objectives for the different trusted communities. We have proposed a 3-level "sensitivity" scale associated to the confidence the enterprise has on this partner community: High, Medium and Low. For each "sensitivity level", we propose to classify information units and processes according to 3 classes: black for highly protected units, grey for units which require standard protection level and White for un-protected units. For example, medical and professional data related to a fireman service are set to "medium" for their own organisational unit and set to high for accesses achieved by other organisational units.

Once the global strategy is set and in order to provide a consistent end-to-end security policy and identify security requirements, we propose a method to propagate security constraints. As the business processes are implemented thanks to a service chain, the first step consists in exposing the service interface including the security requirements (for both service data and service operation). To reach this goal, we have proposed in Deliverable T2.1D2 few rules, related to the process and information "value" (i.e. white, grey or black) to combine the security components.

Add security properties in WS-agreement contract

We propose to set QoP agreements in WS-Agreement by mean of *GuaranteeTerms*, *service level objectives* (SLO), and their related *business values:*

- Agreement terms are an extensible and interoperable means to author and convey the nature of an agreement [38]. AgreementTerm itself is where implementers can specify domain specific terms. These terms can be used to convey meaning during a negotiation exchange or afterwards to characterize an existing agreement. We define an agreement document type and a set of agreement term types that must be used both in creations of an agreement service and in expressing this agreement in an agreement document [38]. Externally defined agreement languages may be used to extend WS-Agreement behaviours in our case we will use NRL-SO terminology to represent "security" terms.
- Service Level Objective: an assertion expressed over service descriptions. each service level objective may have a list of business values attached to it, representing different value aspects of this objective [37]. Both agreement initiator and agreement responder may specify business values. The business value is intended to represent the strength of an agreement in domain-specific terms. In general, business value is an assertion representing a value aspect of a service level objective attached to the service that is the subject of the agreement. The value may be specified in terms of domain-specific qualities such as importance, cost and others.

In the following example (Figure 2-8) we propose an agreement (using our NRL-SO based ontology for specifying security terms see T2.1D1) applied to a service to set the obligation on the service provider to provide SAML authentication protocol AND XML-encryption to encrypt the message.

<wsag:GuaranteeTerm

wsag:Name="SecurityProtocol" wsag:Obligated=”ServiceProvider”>
wsag:ServiceScope="S5">
<wsag:ServiceLevelObjective>
Must_Exist (AuthenticationProtocol_And_EncryptionProtocol)
</wsag:ServiceLevelObjective>
<wsag:BusinessValueList>
<wsag:Preference>
…….
<wsag: AuthenticationProtocol >saml</wsag: AuthenticationProtocol t>
<wsag:EncryptionProtocol>XML-Enc</wsag:EncryptionProtocol >
…….
</wsag:Preference>
</wsag:BusinessValueList>
</wsag:GuaranteeTerm>

Figure 2-8. Example of QoP agreement setting authentication and encryption

Integration of WSDL security annotation

Expressed at a semantic level, security preferences are implemented at the service layer to define the security components to be composed. To express service level security preference e.g. specifying the types of client credentials needed or to specify which parts of the message should be signed and/or encrypted, we need to join a security policy to the service (WSDL) document. Policy references could be made via *policyreference* element as defined in security policy specifications or could be embedded in the WSDL document using policy expression using *wsdl:definitions.*

Hereafter (Figure 2-9) is an example in which a part of the policy attached to WSDL document is used to indicate which parts of the trip reservation service input operation messages are to be encrypted or digitally signed. Lines 4 to 20 indicate that the body and the specified headers need to be signed; lines 22 to 25 indicate that only the message’s body needs to be encrypted.

```
1.  <wsp:Policy wsu:Id="tripReservationInterface_input_message_security_policy">
2.  <wsp:ExactlyOne>
3.  <wsp:All>
4.  <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
5.  <sp:Body/>
6.  <sp:Header Name="To"
7.  Namespace="http://www.w3.org/2005/08/addressing"/>
8.  <sp:Header Name="From"
9.  namespace="http://www.w3.org/2005/08/addressing"/>
10. <sp:Header Name="FaultTo"
11. Namespace="http://www.w3.org/2005/08/addressing"/>
12. <sp:Header Name="ReplyTo"
13. Namespace="http://www.w3.org/2005/08/addressing"/>
14. <sp:Header Name="MessageID"
15. Namespace="http://www.w3.org/2005/08/addressing"/>
16. <sp:Header Name="RelatesTo"
17. Namespace="http://www.w3.org/2005/08/addressing"/>
18. <sp:Header Name="Action"
19. Namespace="http://www.w3.org/2005/08/addressing"/>
20. </sp:SignedParts>
21. <sp:EncryptedParts xmlns:sp=”http://schemas.xmlsoap.org/ws/2005/07/securitypolicy”>
22. <sp:Body/>
23. </sp:EncryptedParts>
24. </wsp:All>
25. </wsp:ExactlyOne>
26. </wsp:Policy>
```

Figure 2-9: Integration of security requirements in a WSDL document

Security ontologies and security pattern building

To exchange precisely the security requirements, a common ontology must be used to precisely describe the semantics of exchangeable security requirements. To fit this goal, NRL-SO (Navy Research Laboratory Security Ontology) covers globally security properties. This ontology is divided into several sub-ontologies [48] devoted to security concepts, resources, algorithms and tools but it needs to be adapted to the service protection context. The solution we propose is structured according to the different protection requirements:

- Transport security: this part is related to the communication infrastructure. Different requirements can be set
    - Either a secured infrastructure is implemented thanks to network or host mechanisms. This secured infrastructure is set before acceding to the service architecture.
    - Or secured communication protocols are used thanks to net security protocols. These protocols are set while communicating messages between nodes
- Message security: this part is divided into two parts:
    - In order to manage access control, we propose to define there authentication and authorisation properties.
    - In order to avoid message repudiation and to control the message identity, digital signature is applied on the message content
- Data security: this part is devoted to the protection of the stored data. This may ead to store encrypted data and to the addition of digital signature on the stored data.
- Environmental security: this last part is mostly devoted to the "process" organisation. It consists in defining access control (i.e. on credential management). This part is related to the credential sub-ontology.
- Authorisation policy: this part refers to access control. As services have to be composed into different service chain, different authorisation policies can be set, from the simplest one to the more constrained one:
    - Access control based to the calling service
    - Access control based on the human user who is "running" the service chain. This access control can be expressed either thanks to a role or thanks to the user identity
    - Access control based on the calling service and on the human user who is "running" the service chain.
    - Access control based on the human user who is "running" the service chain and on the different services invoked in the chain before calling the controlled service.

In order to simplify the security policy and security preferences descriptions, security patterns are built as answers to security objectives defined thanks to a simplified ontology (Figure 2-10), related to the basic security services, fitting the simplified security requirements identification we proposed.

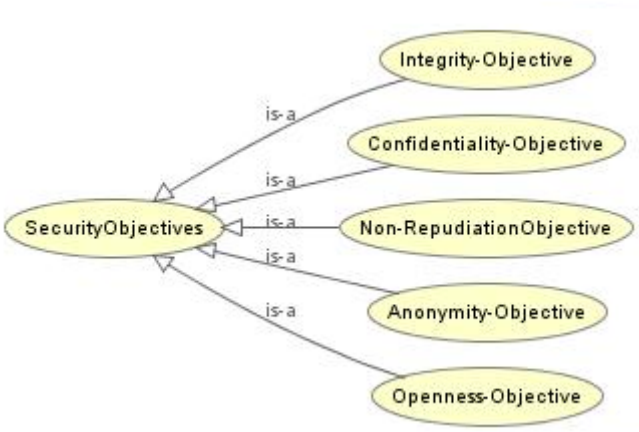


Figure 2-10: Security objectives

These security objectives are then related to security patterns so that the convenient security services can be composed and orchestrated in a convenient way:

- Confidentiality objective: in order to support a end-to-end confidentiality service, the corresponding security pattern includes the following elements:
    - Setting a secured transport system
    - Integration of authorisation policy

- Integrity objective: this objective is mostly related to message exchange and is also related to the non-repudiation objective. To fulfil this requirement, messages must be signed so that the message content and the sender identity can be proved
- Non repudiation objective: this objective is mostly related to message exchange (involving signed messages) but also involves logging the different activities in a stored log file.
- Anonymity objective involves both that the user personal data are protected according to a data privacy agreement (if these data are sent to a service provider) and that a convenient SSO and authorisation components are used to reduce personal information exchange between (may be untrusted) components.
- Openness objective involves that security policy can be expressed in an "opened way". To fulfil this requirement, we propose to define different security reference models to capture the policy semantics (for example while defining role-based access rights).

Integration of the security requirements in the service interface

As proposed in Deliverable T2.1D1, operations are described thanks to YASA4WSDL which can define for each WSDL element two attributes providing semantic description:

- *serviceConcept*, contains a set of URI referencing the corresponding concepts in one or several Technical Ontologies (namely the service ontologies) that describes the semantics of some service concepts: *precondition* and *effect*.
- *modelReference* , contains a set of URI corresponding to the first list and which define the semantics according to one or several Domain Ontologies related to the operation business area and to the non functional property class.

Security requirements are inserted in this description by relating the security ontology concept to the current security requirements. For example, the trip reservation example introduced in T2.1D1 can be enriched by security requirements: secured communications (VPN or SSL based) are required to protect information exchange. This leads to integrate 3 different ontologies: the service ontology, the Domain ontology and the Security ontology. By this way non functional and functional properties are specified in a similar way while registering the service.

Example

In the following example taken from the fire-fighting use case, several security domains are defined (Figure 2-11). The resource allocation process (Figure 2-12) involves several data with different security requirements.

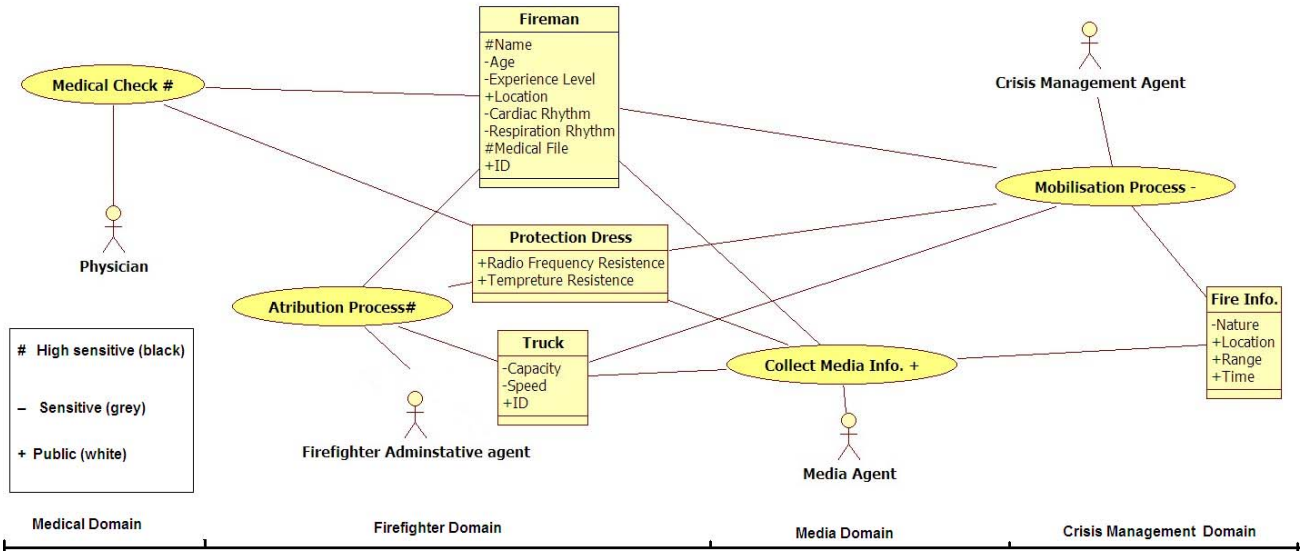


Figure 2-11: Security requirements integrated in the fire-fighting use case

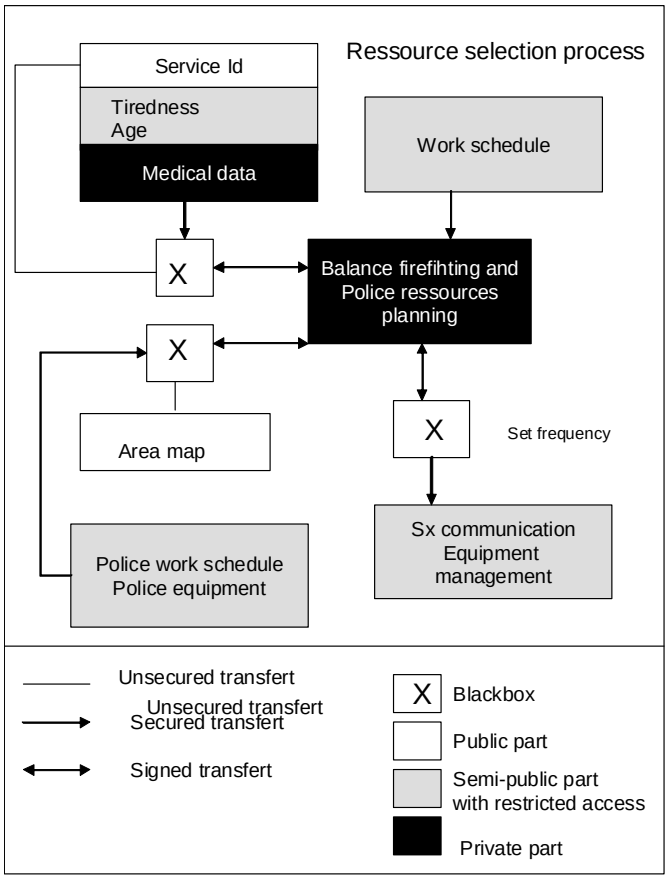


Figure 2-12: Resource allocation process including security annotations

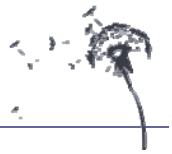
The following figure (Figure 2-13) shows the non-functional contract offered by services augmented by QoP constraints:

S1: non-functional contract	**S2: non-functional contract**
60°C <= Fire resistance <= 90°C Physical strain <= 70 PPM [1] 2Ghz **<** Radio frequency <= 2.5 Ghz **Authentication: Kerberos Protocol** **Transport: HTTPS**	60°C <= Fire resistance <= 90°C 20 <= physical strain <= 70 PPM 2Ghz **<** Radio frequency <= 3Ghz **Authentication: Login** **Transport: HTTP non-secured**
S3: non-functional contract	**S5: non-functional contract**
60°C <= Fire resistance <= 90°C Physical strain <= 100 PPM 2Ghz **<** Radio frequency <= 2.5 Ghz **Authentication: Kerberos Protocol** **Transport: HTTPS**	60°C <= Fire resistance <= 90°C physical strain <= 70 PPM 2Ghz **<** Radio frequency <= 2.5 Ghz **Authentication: SAML** **Transport: HTTPS**

Figure 2-13: Service QoP Constraints in the firefighting use case

Provided a set of functionally filtered services, we are able to retain only the services supposed to provide a compliant QoS at invocation time by running there static non-functional contracts trough the specified QoS constraints:

- Services S1, and S5: the offered contracts exhibit equivalent or stronger QoS as well as QoP constraints than the one required at the process level (according to the sensitivity of the information transmitted). this service will be filtered while S not filtered



- S2 in the other hand has non-secure transport protocol as well as a weak authentication mechanism and thus weaker QoP constraints than those required (according to the sensitivity of the information transmitted) this service will be filtered.
- Service S3: although this service exhibits a suitable quality of protection level, the offered contract exhibits a weaker physical strain constraint than the one required at the process level, consequently, this service will be filtered.

Thus the document describing the security policy of S5 could include the following lines (Figure 2-14).

```
1.  <wsp:Policy xmlns:wsp=".../policy">
2.  <sp:TransportBinding
3.  xmlns:sp=".../securitypolicy">
4.  <wsp:Policy>
5.  <sp:TransportToken>
6.  <wsp:Policy>
7.  <sp:HttpsToken/>
8.  </wsp:Policy>
9.  </sp:TransportToken>
10. </wsp:Policy>
11. </sp:TransportBinding>
12. </wsp:Policy>
13. …
14. <sp:SupportingTokens>
15. <wsp:Policy>
16. <sp:WssSamlV10Token10
17. sp:IncludeToken="... " />
18. </wsp:Policy>
19. </sp:SupportingTokens>
```

Figure 2-14: Example of security policy description

User registry organisation

Organising a secured infrastructure often involves implementing access control functions. Based on user authentication, these access controls can be based on the user identity, a role related to the user activity or to a particular user personal data. In this section we propose to model a user registry so that user-related functional properties (related to its personal data) and preferences can be stored and used by the SEMEUSE framework.

State of the art

Traditionally, user-oriented directory systems (as X500 [52], LDAP[1], …) includes information related to the user authentication (mostly user identity and connection information) as well as a basic "group" description that can be used to describe the user position in the organisation and the role it can play in the different processes he is involved in.

As far as the service architecture is concerned, the OASIS reference model [58] proposes a trust model between services stakeholders and service customers (i.e. participants) in order to define who can perform an action or trigger an event (Figure 2-15). This leads to define several trust level associated with different "social groups" mixing stakeholders and participants and relationships between these domains. In addition, these trust levels are used to define access policies in terms of authorization and authentication strategies so that credentials (integrating roles) and identity (including identifier management) can be used. Trust relationships between participants can also be used to deduce decentralized authentication and authorization mechanisms so that trust chains can be established within a global service chain (Figure 2-16).

[1] LDAP is defined by a set of 10 RFCs : RFC4510 to RFC4519.

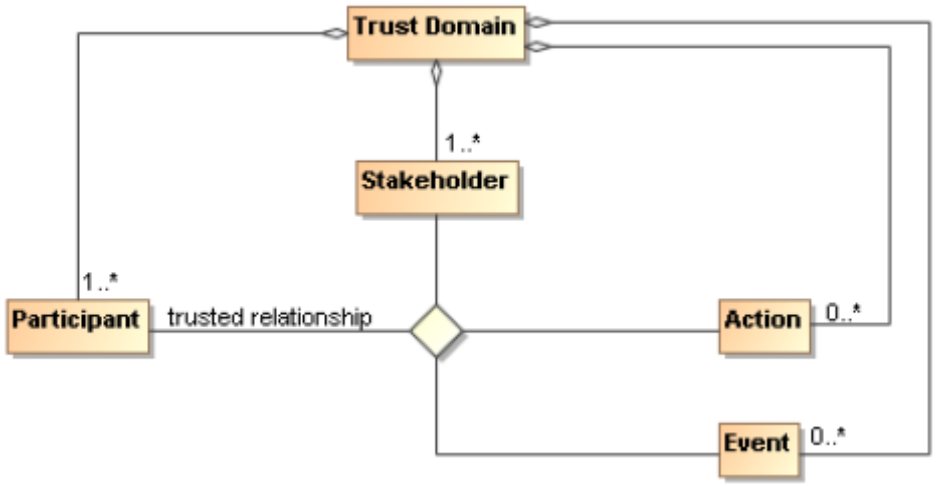


Figure 2-15: Trust model taken from [58] p. 90

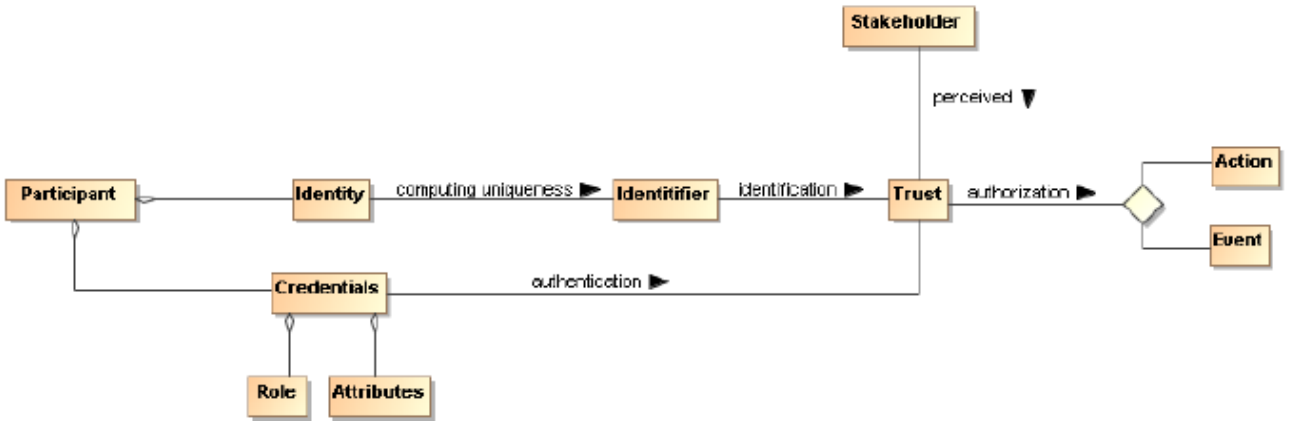


Figure 2-16: Authorisation model taken from [58] p. 90

Trust relationships between participants can also be used to derive decentralised authentication and authorisation so that trust chain can be established for a global service chain (Figure 2-17).

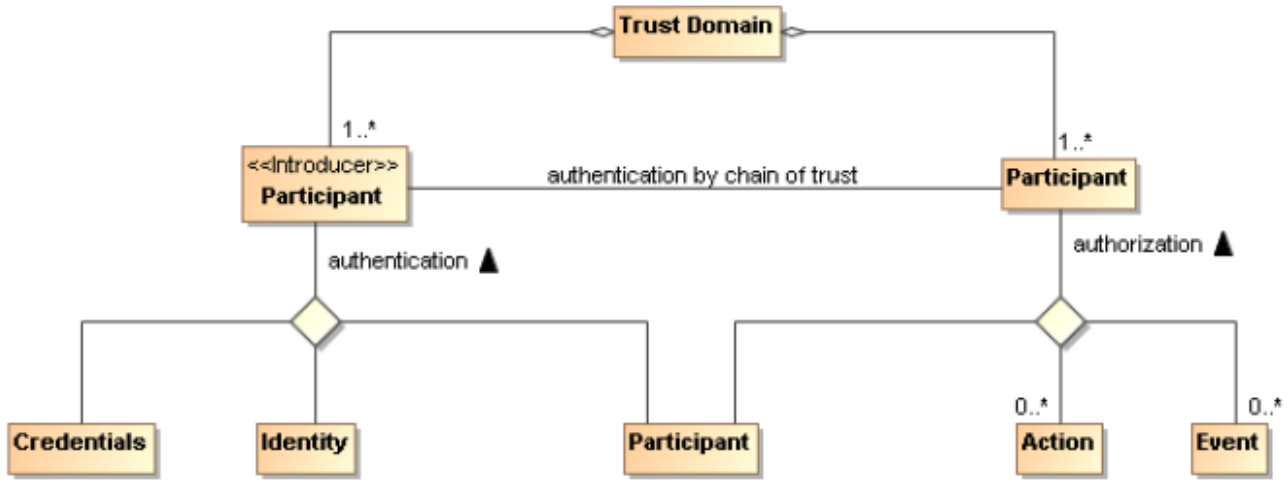


Figure 2-17: Decentralised authorisation model taken from [58] p. 92

Such a credential-based security mechanism expresses security choices that are then used to define security mechanism to enforce security.

Nevertheless this model do not take into account the role or attributes semantic, in order to simplify the access control rule, nor integrate really the customer preferences to set trusted communities.

User registry model

Trusted communities are defined as groups of actors (an actor can be an end-user, a service provider or even a service). The community is defined and owned by an actor who gathers in this group a set of other actors that can be associated to similar security requirements. For example, an end-user can gather purchasing service providers in a “trusted purchasing community” so that services from these providers can be selected and proposed for an electronic payment.

Trust policy is used to describe trust relationships between actors and to define the way communities can be extended (Figure 2-18).

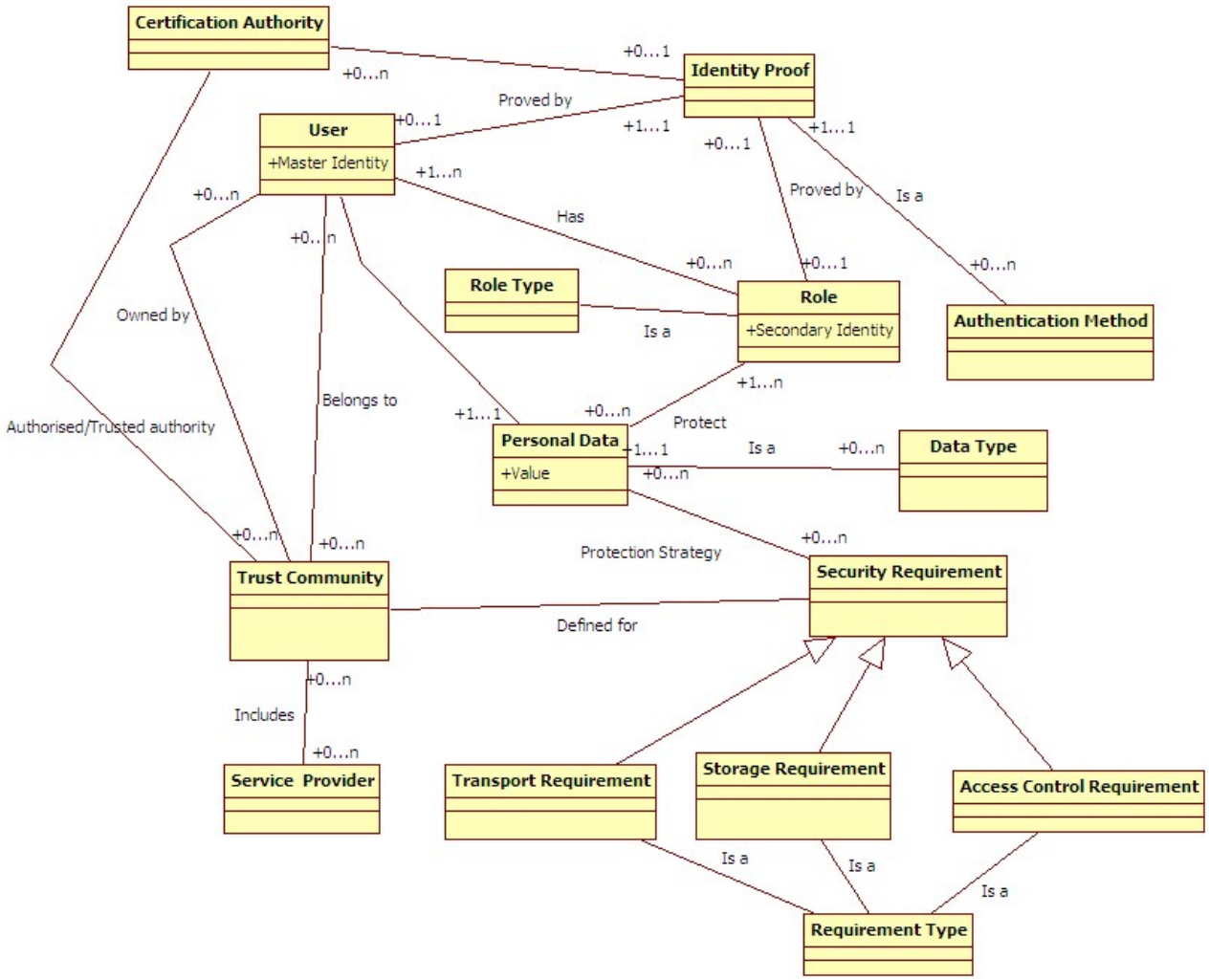


Figure 2-18: Trusted community

Depending on objective and context, a user or a service can play different roles, use several identities and define different QoP sets of requirements according to role, identity and/or context. In our case services include a set of operations and functional properties that can be either produced or consumed by the service. These functional properties are taken either from the provider legacy data or from the consumer personal data. Depending on the data type, the data owner can define security requirements regarding both the way the data is stored, the way access authorisations are given and the way the data is transferred. Role-based access authorisation is used in a major way and nominative access control. Security preferences are expressed both by service providers and service customers. These preferences are used to define security requirements attached to functional property. As services are autonomous units that can be composed in different contexts, security requirements are described for both functional properties and operation contained in a service. The global context is defined by trust communities. A “default” community is used to gather all the unidentified partners. Trust relationships between customers and providers define the current trust level so preferences patterns can be deduced. Trusted communities are defined as groups of actors (i.e. end-user, a service or even a service chain of services invoked by a user initiative). The community is defined and owned by an actor who

gathers a set of other actors that can be associated to similar security requirements. Trust policy describes trust relationships between actors and defines the way communities can be extended.

End-user information can be stored in a particular registry that can form a part of the corporate registry. The data model we propose is described in the figure below (Figure 2-19). The user is defined thanks to his main identity (used as a root point) as a subject in a user registry. Each user can play different roles and each role is used as a key to allow or refuse access. Each role can also be related to a secondary identity and to contextual information describing the motivation (leisure, professional…), the time period associated to the context (if needed) and the trusted communities that can be invoked in this context.

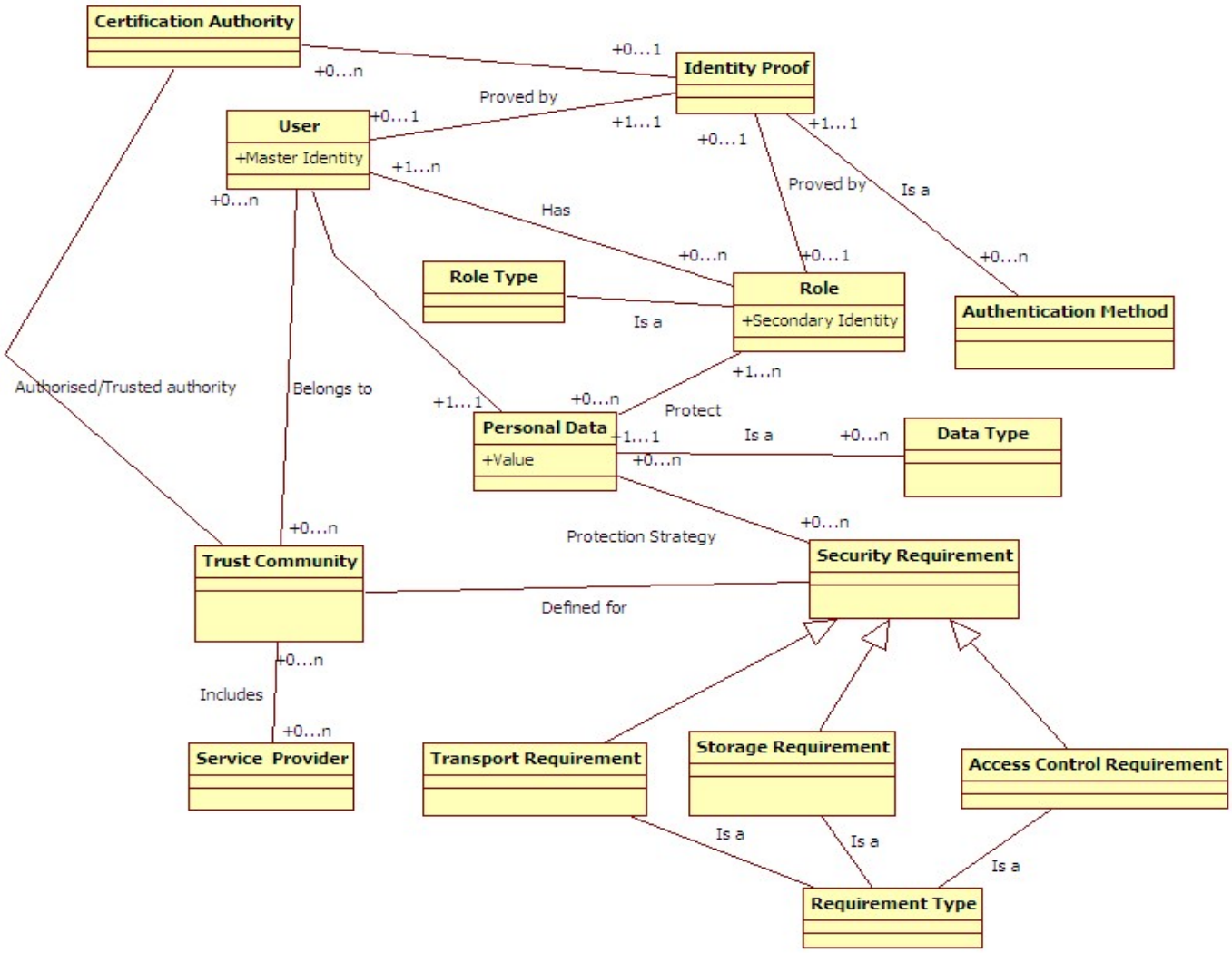


Figure 2-19: User registry data model

User registry integration in the SEMEUSE Framework

In order to fulfil the openness and dependability requirement, the user registry is organised in a distributed way so that nodes can be added, moved from one machine to another… depending on the context. It allows the importation of user information from various nodes (ESB, other directory management systems) and stored the user functional properties: user identity (defined as the subject identity), secondary identifiers (defined as principals) and various attributes including roles (and their related reference models) (Figure 2-20). As proposed in Deliverable T2.2D2, this registry component (see Figure 2-21) includes different sub-components: an IDProvider component is used to manage locally authentication related information, the attribute manger is used to extract the convenient user attribute depending on the authorisation policy whereas the federation management component is used to support the interactions between the distributed user registry instances.

The following figure (Figure 2-22) presents how this component interacts with the PETALS and DRAGON nodes.

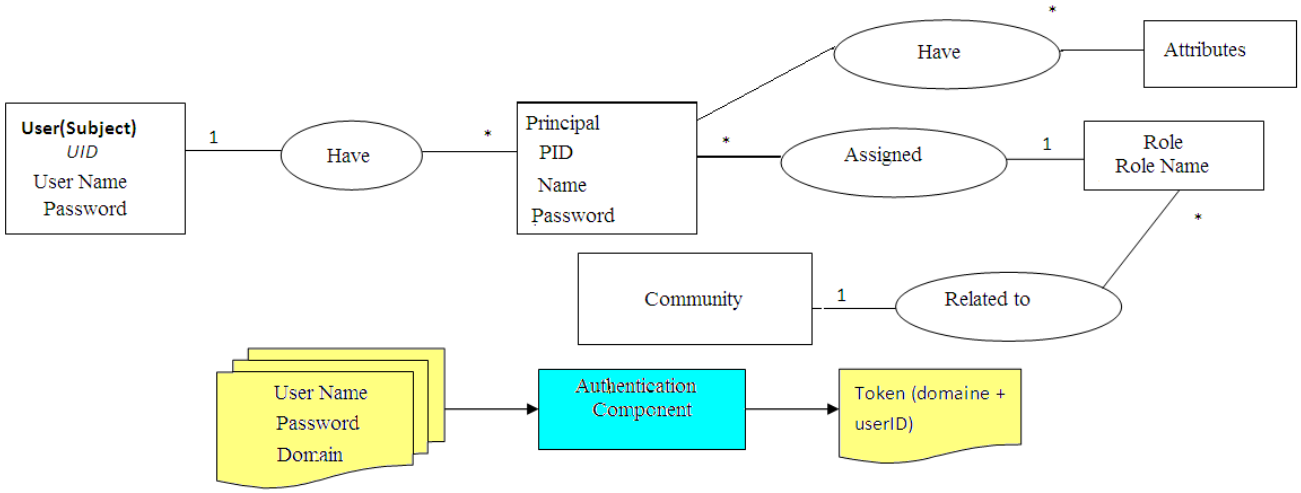


Figure 2-20: SEMEUSE Internal user registry model

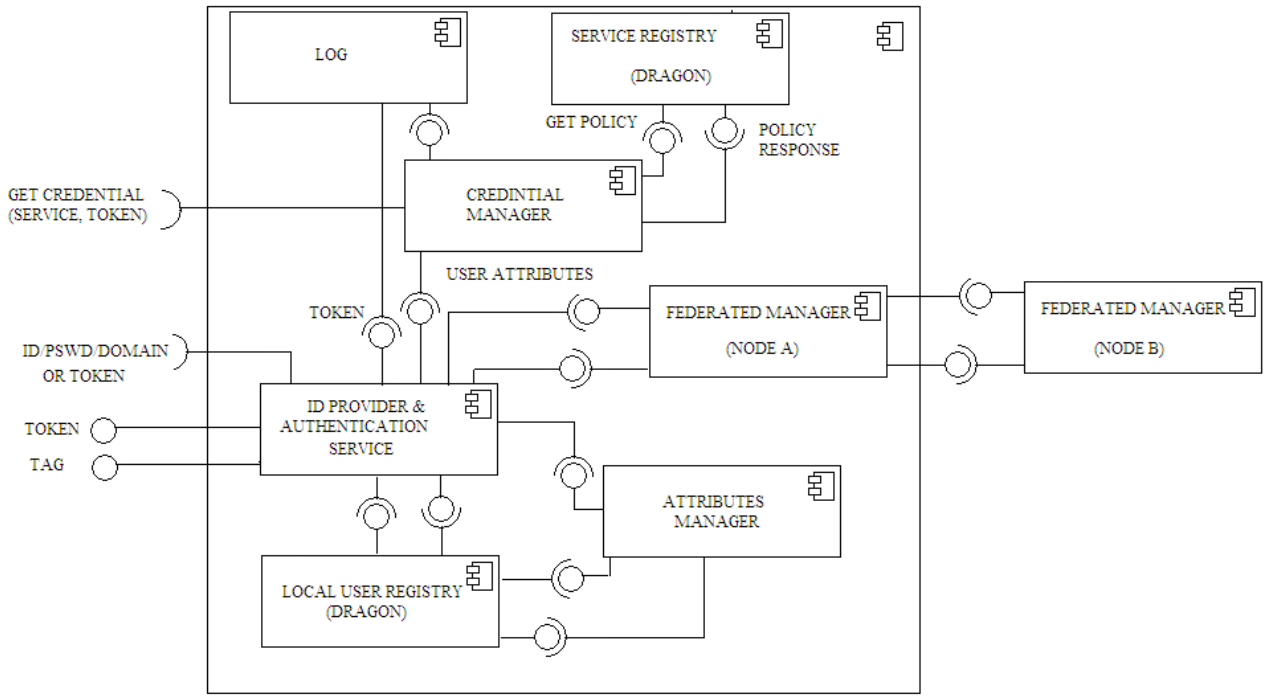


Figure 2-21: Distributed User registry component

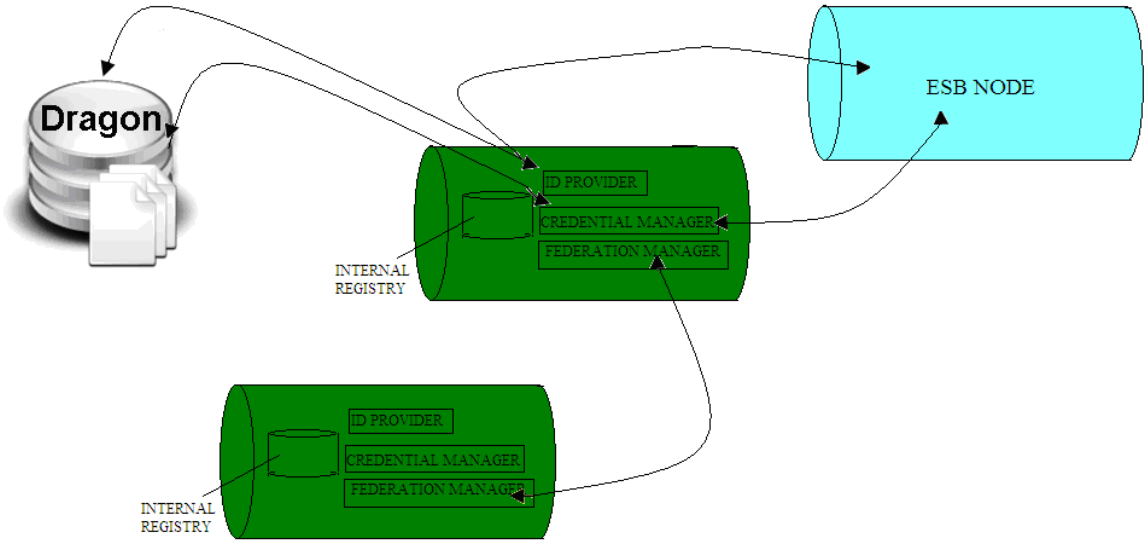


Figure 2-22: Interaction of the distributed user registry with PETALS and DRAGON nodes

Integration of security requirements in the service discovery process

As the security preferences are expressed as semantic annotations using adapted reference model, security concerns can be added in the service discovery in a similar way as functional requirements.

We assume that each user has logged once in the system before defining a service discovery request. Thanks to the Single Sign On system, a token including the reference to the node hosting the user information is associated to the user so that its security preferences and security-related attributes can be extracted easily.

The first step consists in extracting the user security preferences and the different roles he can play. These user functional properties are added in the basic query. As the service semantic description embeds the semantic security annotation (including role based controlled authorisations), these user related information can be added easily as in the “functional requirements” (Figure 2-23).

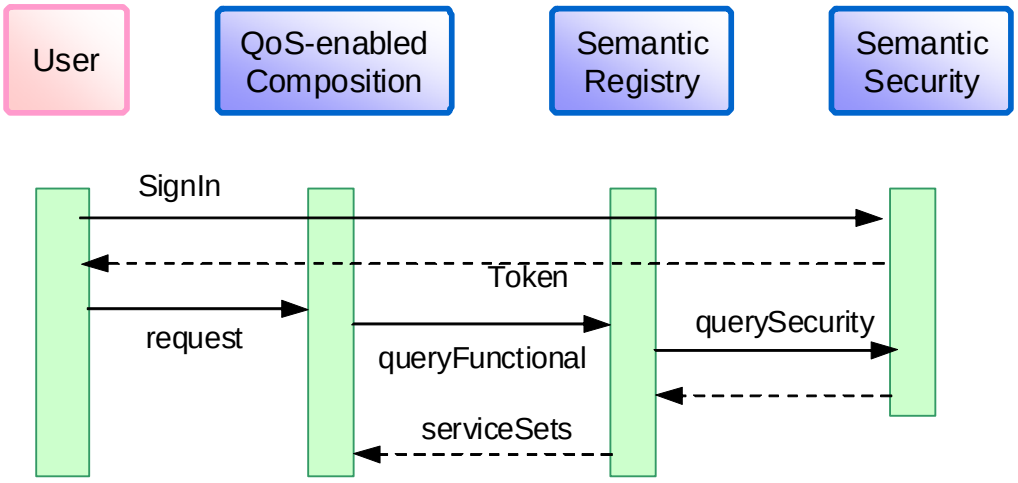


Figure 2-23: Selection process integrating security requirements

As different matching policies can be set, we have to control more precisely if the service can be invoked or not by the user (as partial matching is allowed in the discovery process). This requirement involves invoking the authorisation service (see Deliverable T2.2D2 and the following Figure 2-24) in order to confirm / infirm if the service can be selected or if it has to be removed from the discovered service set.

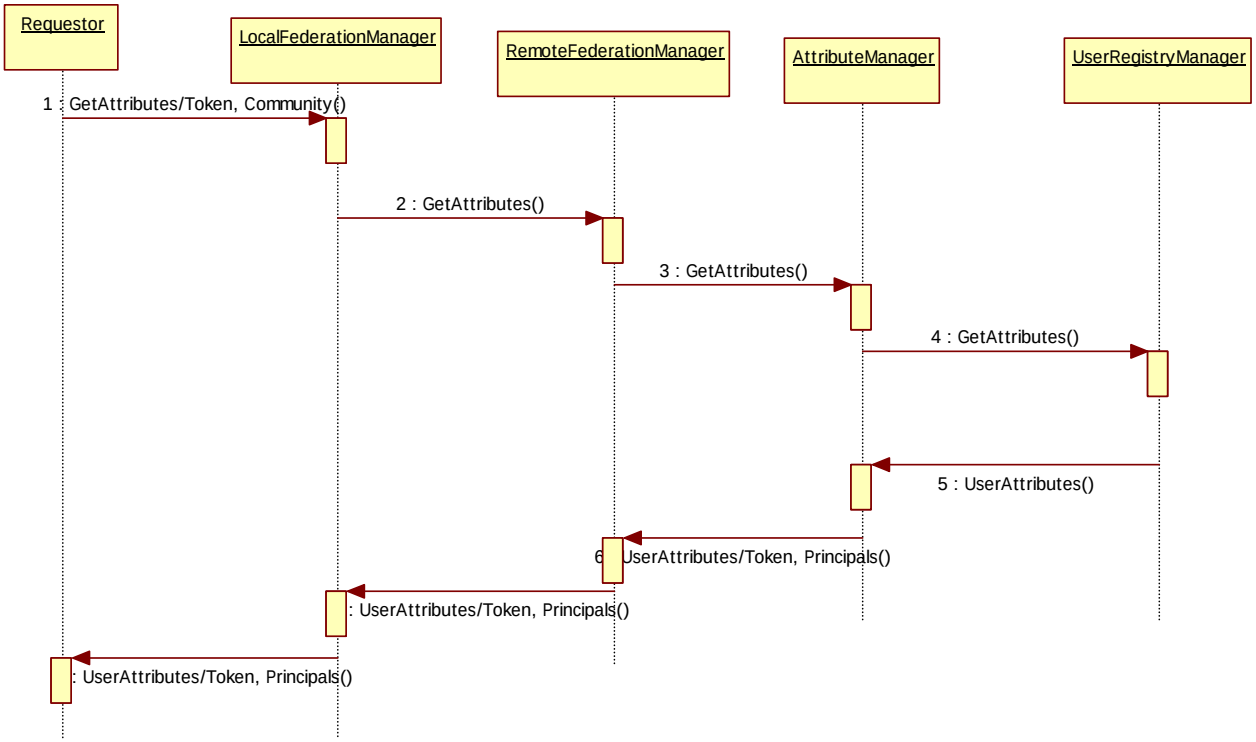
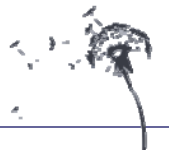


Figure 2-24: Distributed authorisation process



3. Service discovery

Service discovery is based on matching between the user task description (required) and services description (provided) presented in the previous chapter. Matching is done with regard to functional, QoS and security properties. We present the service registry and the associated functional matching in Deliverable T2.3D1 “Semantic Registry”. We plan to further study and determine the way in which functional, QoS and security matching/filtering are applied (e.g., in which order), so that optimal performance can be achieved.

4. Service composition satisfying global QoS requirements

4.1. QoS-aware Service Composition in Dynamic Service Oriented Environments

Service Oriented Computing (SOC) and its underlying technologies such as Web Services have emerged as a powerful concept for building software systems [17]. An interesting feature of SOC is that it provides a flexible framework for reusing and composing existing software services in order to build value-added service compositions able to fulfill complex tasks required by users. A key requirement in services' composition is to enable these tasks while meeting Quality of Service (QoS) constraints set by users.

QoS-aware service composition underpins this purpose since it allows for composing services able to fulfill user required tasks while meeting QoS constraints. Assuming the availability of multiple resources in service environments, a large number of services can be found for realizing every sub-task part of a complex task. A specific issue emerges to this regard, which is about selecting the best set of services (i.e., in terms of QoS) to participate in the composition, meeting users' global QoS requirements

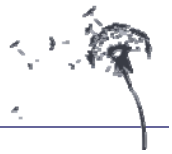
QoS-aware composition becomes even more challenging when it is considered in the context of dynamic service environments characterized with changing conditions. The dynamics of service environments bring about two specific problems in service selection: First, as dynamic environments call for fulfilling user requests on the fly (i.e., at run-time) and as services' availability cannot be known a priori, service selection and composition must be performed at runtime. Hence, the execution time of service selection algorithms is heavily constrained, whereas the computational complexity of the problem is NP-hard. The second issue is about the fluctuation of QoS conditions due to the dynamics of such environments. This problem arises for example when one or more services that make part of a service composition are no longer available or their QoS decreases (e.g., due to network disconnection or weak network connectivity) during the execution of the composition. Thus, a service selected to participate in a composition based on its QoS may no longer provide the same QoS when the time comes to be actually invoked. The overall question asked to this regard is: How to cope with the dynamics of service environments during the selection, the composition and the execution of services?

In this chapter, we present a service selection algorithm that copes with the above issues. Our algorithm is centered on *dynamic binding* of services, i.e., binding one out of multiple possible services just-in-time before its invocation according to its QoS measured at runtime (hereafter referred to as *runtime QoS*). Our selection algorithm underpins this purpose since it selects multiple services for every sub-task part of a complex task required by users, based on their nominal QoS (hereafter referred to as *advertised QoS*). Our algorithm consists in a guided heuristic. Our choice of a heuristic-based approach replies to the two issues stated above for dynamic environments: First, since the time available for service selection is limited, brute-force-like algorithms are inappropriate for such purpose, as they target determining the optimal composition, which is NP-hard. Second, finding the optimal composition may prove useless in the end since, due to dynamics, there is no guarantee that the selected composition will be possible at runtime or that its runtime QoS will not decrease with respect to the advertised one. To this regard, our algorithm aims at determining a set of alternative service compositions that maximize QoS utility and their runtime QoS may respect global QoS constraints. At runtime, only one selected, and if it is no longer possible or its QoS decreases, an alternative composition will be executed.

4.2. Related Work

Several selection algorithms have been proposed to select service compositions with different composition structures and various QoS constraints. A taxonomy of these solutions may be produced based on their objectives and the way they proceed. According to this, a first class of approaches aims at determining the optimal service composition (i.e., composition with the highest QoS utility) using brute-force-like algorithms (e.g., Global Planning [20], BBLP, MCSP, WS-IP [21]).

Such solutions have high computational cost and they cannot provide a solution in a satisfying amount of time, thus they are inappropriate to be used in the context of dynamic service environments. To cope with this issue, other approaches propose heuristic-based solutions (e.g., WS-HEU and WFlow [21], Genetic algorithm [22,23]) aiming to find near-optimal compositions, i.e., compositions that respect global QoS constraints and maximize a QoS utility function. Yu et al. [21] present two heuristics, WS-HEU and WFlow, for the service selection problem. WS-HEU is specific heuristic applied to sequential workflows (i.e., workflows structured as a sequence of activities), whereas WFlow is designed for general workflow structures (i.e., sequential, conditional, parallel). The main idea of WFLow is to decompose workflows into multiple execution routes.



WFlow considers a parameter _i for every route indicating its probability to be executed. Therefore, it focuses on the route with the highest probability, whereas in our approach we aim at giving feasible service compositions regardless of the way the workflow will be executed.

Other approaches [22,23] present heuristics based on the genetic algorithm. Such approaches aim at selecting the optimal composition. Although it produces satisfying results in terms of the quality of the selected composition, the application of the genetic algorithm to the service selection problem presents many drawbacks: First, the order in which service candidates are checked is randomly chosen (e.g., Crossing [22]), whereas in our approach we aim at checking services in an ordered way to optimize the timeliness and the accuracy of our algorithm. Second, as the genetic algorithm can run endlessly, the users have to define a constant number of iterations fixed a priori. However, fixing a high number of iterations does not give any guarantee about the quality of the result. Third, the genetic algorithm aims at selecting only one composition rather than a set of compositions. Therefore, it is deemed non useful for our purpose (i.e., selecting many compositions to enable dynamic binding of services).

More recently, Alrifai et al. [24] presented a novel approach that combines local and global optimization techniques. This approach starts from the global level and resolves the selection problem at the local level, i.e., it proceeds by decomposing global QoS constraints (i.e., imposed by the user on the whole composition) into a set of local constraints (i.e., for individual sub-tasks, part of the whole composition). To do so, it uses MILP (mixed integer linear programming) techniques to find the best decomposition of QoS constraints. The main drawback of this approach is that the decomposition of QoS constraints is static (i.e., based on fixed levels of QoS), and thus it discriminates a number of service candidates.

4.3. Composition Approach Overview

Our approach starts from the assumption that the user uses a Graphical User Interface to submit his/her request. The interface guides the user to express his request in terms of functional and QoS requirements, and then it formulates these requirements as a machine-understandable specification.

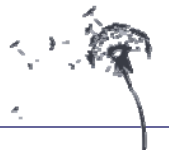
User functional requirements are formulated as an abstract task (hereafter referred to as *abstract service composition*) brought about by the composition of a set of abstract sub-tasks (hereafter referred to as *activities*). These activities are described with abstract information (i.e., function, I/O description). Abstract service compositions are later transformed into *concrete service compositions* by assigning one or more concrete services to every activity in the composition. These services are selected among a larger set of services initially considered as eligible to fulfil the activity while meeting its requirements. We refer to these eligible services as *service candidates* of the considered activity.

Concerning user QoS requirements, they are formulated as a set of constraints (hereafter referred to as *global QoS constraints* on the whole composition. These constraints cover several QoS attributes specified by the user. The details about QoS attributes are given by the QoS model presented in Section 2.2.1.

Once user requirements are specified, we proceed by automatically building executable service compositions with respect to user requirements and the dynamics of the service environment. Building executable compositions consists of: (i) discovering, (ii) selecting, and (iii) composing services on-the-fly (i.e., at runtime).

Services' discovery is based on the semantic approach outlined in Chapter 3. This approach uses functional, QoS and security ontologies to match user requirements to services available in the environment. The matching is based on an efficient semantic reasoning performed at runtime. For every activity in the composition, the discovery phase gives the set of service candidates able to meet the functional, QoS and security requirements. From a QoS point of view, services' discovery uses advertised QoS of services to perform a preliminary filtering ensuring that individual service candidates respect user QoS requirements.

Refining the first filtering, the selection phase ensures user QoS requirements at the global level (i.e., for the whole composition) based on the advertised QoS of services. That is, it selects a set of service candidates for each abstract activity that, when composed together, meet global QoS constraints. To achieve this, we introduce a heuristic algorithm based on clustering techniques, notably K-Means [27]. Clustering techniques, applied to our purpose, allow for grouping services with respect to their QoS into a set of clusters, to which we refer as *QoS levels*. Each level includes the set of services having approximately the same QoS, and it is characterized with a centroid that represents the mean QoS of all services. The centroids associated to all the activities are further composed together to determine which combination of QoS levels can meet global QoS requirements and yield the highest QoS utility. The importance of this approach is that it reduces the number of combinations to consider since it checks the feasibility of compositions based on the centroids instead of services.



More specifically, our algorithm deals with the service selection problem in two phases: (1) a local classification phase, which clusters service candidates into a set of QoS levels and determines a utility value for each level; this phase is performed for every activity in the composition; (2) a global selection phase which selects the set of QoS levels meeting global QoS requirements and yielding the highest QoS. This phase uses a heuristic guided by the QoS levels' utilities.

Once the global selection is fulfilled, the composition phase uses service candidates associated to the selected QoS levels to define an executable and adaptive service composition. It proceeds by replacing every abstract activity in the composition with a 'dynamic binding' activity that takes as input the set of service candidates associated to the QoS level selected for the considered activity. At runtime, a unique service is selected and enacted among the provided ones with respect to its runtime QoS.

4.3.1. QoS Model

We consider the QoS model presented in Section 2.2.1. Our model allows for specifying cross-domain QoS attributes like response time, cost, availability, reliability, as well as domain-specific QoS attributes. Our model provides a detailed taxonomy of QoS which is flexible and easily extendible. Herein, we introduce an extension that concerns a particular classification of QoS attributes needed for our composition approach. QoS attributes can be divided into two groups: quantitative attributes (e.g., response time, availability) and qualitative attributes (e.g., security, manageability [28]). The former attributes are quantitatively measured using metrics, whereas the latter attributes cannot be measured, they are rather evaluated in a boolean manner (i.e., they are either satisfied or not). For the sake of simplicity and without loss of generality, in this work we will consider only quantitative QoS attributes. The latter attributes are in turn divided into two classes: negative attributes (e.g., response time, cost) and positive attributes (e.g., availability, reliability). The first class of attributes has a negative effect on QoS, (i.e., the higher their values, the lower the QoS), hence they need to be minimized. On the contrary, positive QoS attributes need to be maximized, since they increase the overall QoS (i.e., the higher their values, the higher the QoS).

On the other hand, QoS attributes' values are determined in two ways: During the selection of services, these values are given by service providers (e.g., based on previous executions of services or using users' feedback). As already stated, we refer to these values as advertised QoS, which is specified in services' descriptions. At runtime, QoS values are provided by a monitoring component to enable further dynamic evaluation of services. As already stated, we refer to these values as runtime QoS.

4.3.2. Composition Model

Our algorithm aims at determining a set of near-optimal compositions. Such purpose requires evaluating the QoS of possible service compositions with respect to their structure and the way QoS is aggregated. That is, the evaluation of QoS depends on the structuring elements used to build the composition, to which we refer as composition patterns, and also QoS aggregation formulas associated to each pattern. Next, we describe the composition patterns on which our approach is based and we give the aggregation formulas associated to QoS attributes and composition patterns.

Composition Patterns. We consider a set of patterns commonly used by composition approaches [20,21], which cover most of the structures specified by composition languages (such as BPEL) [29,30]:
- *Sequence*: sequential execution of activities
- *AND*: parallel execution of activities
- *XOR*: conditional execution of activities
- *Loop*: iterative execution of activities

Computing the QoS of Composite Services. For every activity in the abstract service composition, we represent the QoS of a single candidate service S_i by using a vector $\mathbf{QoS_{Si} = <q_{i,1},\ldots,q_{i,n}>}$, where n represents the number of QoS attributes required by the user and $\mathbf{q_{i,j}}$ represents the value of the QoS attribute **j (1 < j < n).** The QoS of a service composition is evaluated based on the QoS vectors of its constituent services while taking into account the composition patterns. Regarding QoS associated with AND and XOR, we adopt a pessimistic approach that considers worst-case QoS values. That is, to determine the values of the QoS attributes of a service composition, we consider the worst QoS values of all the possible executions of the composition. For instance, to determine the response time of parallel activities (i.e., AND), we consider the activity with the longest response time. Concerning the particular case of iterative activities (i.e., structured as a loop), we adopt a history-based estimation that considers the maximum number of loops (i.e., pessimistic approach). This number is determined from previous executions of the activity. In Table 4-1, we show examples of QoS computation with respect to QoS

attributes and composition patterns. These examples can be classified as negative attributes (e.g., response time) and positive attributes (e.g., reliability, availability).

QoS attributes	Composition Patterns			
	Sequence	AND	XOR	Loop
Response time (RT)	$\sum_{i=1}^{n} rt_i$	$max(rt_i)$	$max(rt_i)$	$rt \times k$
Reliability (RE)	$\prod_{i=1}^{n} re_i$	$\prod_{i=1}^{n} re_i$	$min(re_i)$	re^k
Availability (AV)	$\prod_{i=1}^{n} av_i$	$\prod_{i=1}^{n} av_i$	$min(av_i)$	av^k

Table 4-1. QoS computation examples: rti, rei, avi represent respectively, response time, reliability and availability of services candidates structured with respect the composition patterns, whereas RT, RE, AV represent the aggregated values of response time, reliability and availability, respectively.

Notations. To state the problem that we are addressing in a formal way, we use the following notations:

- **AC = <A1,…,Ax>** is an abstract service composition with x activities.
- **CC = <S1,…,Sx>** is a concrete service composition with x service candidates, every service candidate **Si** is bound to an abstract activity **Ai (1 ≤ i ≤ x).**
- **U = <U1,…,Un>** is a set of global QoS constraints imposed by the user on n QoS attributes.
- QoS of a service candidate **Si** is represented as a vector **QoS_{Si} = <$q_{i,1}$,…,$q_{i,n}$>** where **$q_{i,j}$** represents the advertised QoS of the attribute **j (1 ≤ j ≤ n).**
- QoS of a composition **C** is represented as a vector **QoS_C = <Q_1,…,Q_n>** where Q_j is the aggregated value of QoS attribute **j (1 ≤ j ≤ n).**
- Each service candidate **Si** has an associated utility function **fi.**
- Each composition **C** has an associated utility function **F**.

4.4. Service Selection Algorithm

4.4.1. Design Rationale

In the literature, service selection algorithms fall under two general approaches: (i) local [4] and (ii) global selection [21]. The former approach proceeds by selecting the best services (in terms of QoS) for every abstract activity individually. This approach has a low computational cost but it does not guarantee meeting global QoS constraints imposed by the user on the whole composition. Conversely, global selection ensures meeting global QoS constraints since it selects the optimal service composition, i.e., a composition that respects global QoS constraints and has the highest QoS. Nevertheless, the computational cost of global selection is NP-hard. To meet global QoS constraints in a timely manner, we present a heuristic algorithm that combines local and global selection approaches.

The local selection starts from the idea that service candidates associated to a given activity can be grouped into multiple QoS levels (i.e., clusters) with respect to their QoS. Each level contains the set of service candidates having roughly the same QoS. The QoS levels are determined by their centroids whereas the QoS of a centroid is computed as the average of all QoS values of its associated service candidates. Such classification is further used to improve the timeliness of the global selection phase. Indeed, instead of checking the feasibility of service compositions by considering combinations of services (i.e., which leads to the combinatorial explosion problem), we can check the feasibility of service compositions by considering the combinations of QoS levels (i.e., centroids), which reduces significantly the number of combinations to be considered by our algorithm. Our algorithm selects a composition of QoS levels (i.e., one level per abstract activity) that respects user QoS requirements and maximizes QoS. It yields as output an adaptive service composition that gives for every activity, the set of service candidates associated to the selected QoS level.

Although, it is not as accurate as the selection based on QoS of services (i.e., since QoS values of service candidates are slightly different from QoS values of their centroids), this approach can lead to significant enhancements in the timeliness of the algorithm. Additionally, in our approach we improve the accuracy by increasing the number of QoS levels (i.e., clusters), which reduces the range of QoS values, and hence the

QoS of the centroids is closer to QoS of their associated services, thus increasing the accuracy of the selected compositions.

Additionally, in the context of dynamic environments, the issue related to the accuracy exists any way as services' QoS may vary at runtime. Actually, there is no guarantee that the composition selected based on services' QoS values (i.e., in opposition to composition selected based on QoS levels' values) remains possible at runtime.

Starting from the assumption that service candidates (for every activity in the abstract process) are already given by the semantic discovery phase, our algorithm proceeds through the following phases:

1. **Scaling phase**, which is a pre-processing phase aiming to normalize QoS values associated to negative and positive QoS attributes;
2. **Local classification**, which aims at classifying candidate services (for every activity in the abstract process) according to different QoS levels; this classification is further used to determine the utility of every QoS level regarding the global selection phase;
3. **Global selection**, which aims at using the obtained utilities to guide selecting the set of QoS levels that, when composed together, meet user QoS requirements and maximize QoS utility.

4.4.2. Scaling Phase

As already mentioned, QoS attributes can be either negative or positive, thus some QoS values need to be minimized whereas other values have to be maximized. To cope with this issue, the scaling phase normalizes every QoS attribute value by transforming it into a value between 0 and 1 with respect to the formulas below [20].

$$\text{Negative attributes}: \quad q'_{i,j} = \begin{cases} \frac{q_j^{max} - q_{i,j}}{q_j^{max} - q_j^{min}} & \text{if} \quad q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{else} \end{cases} \qquad (1)$$

$$\text{Positive attributes}: \quad q'_{i,j} = \begin{cases} \frac{q_{i,j} - q_i^{min}}{q_j^{max} - q_j^{min}} & \text{if} \quad q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{else} \end{cases} \qquad (2)$$

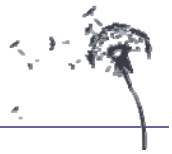
where $q'_{i,j}$ denotes the normalized value of QoS attribute j associated to service candidate S_i. It is computed using the current value $q_{i,j}$ and also $q_{max,j}$ and $q_{min,j}$, which refer respectively to the maximum and minimum values of QoS attribute j among all service candidates.

The same formulas are also used to normalize the aggregated QoS values of compositions, each composition **C** is represented by a vector $QoS_c = \langle Q_1,\ldots,Q_n \rangle$ with **n** QoS attributes. The normalization produces a QoS vector $QoS'_c = \langle Q'_1,\ldots,Q'_n \rangle$. The values of Q'_j $(1 \leq j \leq n)$ are computed based on the current value Q_j, and also $Q_{max,j}$ and $Q_{min,j}$, which refer respectively to the maximum and minimum values of Q_j among all compositions.

4.4.3. The Local Classification Phase

Local classification is performed locally for every activity in the abstract service composition. It aims at grouping service candidates of a given activity into several QoS levels using clustering techniques. It further uses the clustering results to determine a utility value for every QoS level indicating its relative importance regarding services' selection. Indeed, we aim at selecting service compositions that maximize QoS and may respect QoS requirements at runtime, but also selecting a number of compositions as large as possible. Indeed, the larger the number of selected compositions is, the larger is the choice of services allowed during dynamic binding. Additionally, providing a large number of compositions helps preventing the starvation problem during dynamic binding of services. This problem arises when, e.g., a small number of services are selected for dynamic binding but none of them is available at runtime. To this regard, the importance of a QoS level is determined by its QoS values and also the number of its associated services. Next we sketch the main steps towards the computation of QoS levels and their utilities.

Classification Overview. We use the K-means algorithm [11] to cluster service candidates of a given activity. K-means provides a simple and efficient way to classify a set of data points into a fixed number of clusters. These data points are characterized by their coordinates **(x, y)**. The main idea of K-means is to define a centroid $Cen(x_c,y_c)$ for every cluster and to associate each data point $dp_i = (x_i,y_i)$ to the



appropriate cluster by computing the shortest Euclidian distance D between the data point and the centroids:

$$D = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} \qquad (3)$$

Further, the values of centroids are updated by computing the average of their associated data points. The clustering iterates by alternating these two steps (i.e., updating centroids, clustering data points) continuously until reaching a fixpoint (i.e., centroids' values do not change any more). The result of K-means will be the set of final clusters and their associated data points. However, it is worth noting that K-means has a polynomial computational cost in function of the number of iterations [15]. In our context, we use K-means to group service candidates of every activity in the abstract service composition into multiple QoS levels. QoS levels are thus represented as clusters and service candidates are considered as data points determined by their QoS values. Nevertheless, K-means is meant to be applied to data points with two dimensions (i.e., coordinates **(x,y)**), whereas services are characterized by multiple QoS attributes. To cope with this issue, we use K-Means extended to n dimensions (where n is the number of QoS attributes) [32] and we represent data points with QoS vectors $\mathbf{QoS_{Si} = <q_{i,1},\ldots,q_{i,n}>}$. The distance formula is then a N-dimensional Euclidian distance:

$$D = \sqrt{\sum_{j=1}^{n} (q_{c,j} - q_{i,j})^2} \qquad (4)$$

QoS Levels Computation. To cluster service candidates, we need first to determine the initial values of QoS levels (i.e., centroids). For this matter, we define m QoS levels (i.e., $\mathbf{QL_l}$, $\mathbf{(1 \le l \le m)}$), where **m** is a constant number fixed a priori. The value of **m** can be fixed by a domain expert (i.e., expert in the business domain covered by the services) with respect to the service density in the environment and the number of the services needed to cope with the runtime issues.

The value of each QoS level is determined by dividing the range of the **n** QoS attributes (fixed by the global QoS constraints) into **m** equal quality ranges $\mathbf{q_r}$ with respect to the following formula [8]:

$$qr_j^l = \begin{cases} q_j^{min} & l = 1 \\ qr_j^{l-1} + \frac{q_j^{max} - q_j^{min}}{m-1} & 1 < l < m \\ q_j^{max} & l = m \end{cases} \qquad (5)$$

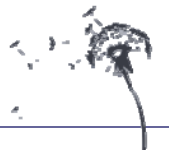
where $\boldsymbol{qr_j^l}$ denotes the quality range **l** of QoS attribute **j** with (**1 ≤ j ≤ n**), whereas **qmax,j** and **qmin,j** refer to the maximum and minimum values of the attribute **j**, respectively. The initial value of each QoS level is then: $\mathbf{QL_l = <qr_1^l, \ldots, qr_n^l>}$ with **1 ≤ l ≤ m**. Once the initial values of QoS levels are determined, we perform the clustering of service candidates to obtain the final set of QoS levels.

Utilities Computation. As already explained, the utility $\mathbf{f_l}$ of a QoS level $\mathbf{QL_l}$ is determined by two parameters: (i) QoS of $\mathbf{QL_l}$ and (ii) the number of its associated service candidates. The first parameter shows that the higher QoS of $\mathbf{QL_l}$ is, the higher is the probability of its associated service candidates to be part of feasible compositions. Concerning the second parameter, if the number of service candidates associated to $\mathbf{QL_l}$ is large, this means that using $\mathbf{QL_l}$ would eventually lead to finding more feasible compositions. Therefore, f_l is computed as follows:

$$f_l = (l/m) * (r/t) * qos_l \qquad where \quad qos_l = (\sum_{j=1}^{n} q'_{i,j})/n \qquad (6)$$

Where:

- **r** is the number of service candidates associated to $\mathbf{QL_l}$ and **t** is the total number of service candidates for the activity, thus the factor **r/t** represents the percentage of services associated to $\mathbf{QL_l}$.
- $\mathbf{qos_l}$ is the QoS utility of service $\mathbf{QL_l}$. It is computed as the average of the normalized QoS attributes' values $\mathbf{q'_{i,j}}$.
- **l** is the level of $\mathbf{QL_l}$ whereas **m** is the highest level, the factor **l/m** is used to decrease the utility of the lower levels**.**



The value of f_I is comprised between 0 and 1 as the values (**l/m**), (**r/t**) and qos_I are also between 0 and 1.

4.4.4. The Global Selection Phase

Global selection aims at selecting QoS level compositions that (i) respect global QoS constraints and (ii) maximize the utility function **F**. The utility function **F** of a QoS level composition **C** with a QoS vector $QoS_C=<Q_1,\ldots,Q_n>$ is defined as the average of its normalized QoS values Q'_j **(1 ≤ j ≤ n)**:

Therefore, the problem that we are addressing can be stated as finding concrete service compositions that fulfill these two conditions:

1. For every QoS attribute **j (1 ≤ j ≤ n)**,
    a. $Q_j \leq U_j$ for negative attributes;
    b. $Q_j \geq U_j$ for positive attributes;
2. The QoS utility **F** is maximized.

Heuristic Overview. The goal of our heuristic is to use the utilities f_I resulting from the local classification phase to select near-optimal compositions without considering all possible combinations of QoS levels. Towards this purpose, we fix a utility threshold **T** that allows for considering only QoS levels with a utility value $f_I \geq T$, thus enabling to focus on the most eligible QoS levels (i.e., QoS levels with the highest f_I values).

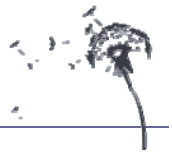
The choice of the threshold **T** is of great importance in our algorithm since it allows for tuning the trade-off between the quality and the timeliness of the algorithm. Indeed, if **T** increases, the number of considered QoS levels possibly decreases and consequently so will the number of compositions to check. Hence, the execution time of the algorithm decreases, but the utility of the resulting near-optimal composition may decrease as well, since the algorithm may discard feasible compositions that provide better utilities. Conversely, if **T** decreases, the number of QoS levels to consider possibly increases and so will the number of compositions to check. Hence, a better utility may be achieved, however the execution time of the algorithm increases as well.

The latter point leads to another important result, which is about the application of our algorithm. Indeed, tuning **T** makes our algorithm generic and flexible, so that it can be applied to multiple dynamic service environments according to their characteristics, particularly their service density [28] and also time constraints in such environments. For instance, if a service environment has a high service density (i.e., hence a large number of QoS levels), **T** can be tuned so that the algorithm will be more selective and checks a limited number of QoS level compositions. By the same, if the execution time in dynamic service environments is heavily constrained (e.g., highly dynamic environments), **T** can be also tuned to make the algorithm check a limited number of QoS level compositions, thus enabling to respect the time constraints of such environments.

Pruning the Search Tree. Our algorithm proceeds by exploring a combinatorial search tree built from QoS levels according to the following rules:

- Every QoS level QL_I having $f_I \geq T$ is a node in the search tree;
- If there is a link (i.e., control flow) from activity A_x to activity A_y in the abstract composition, then the selected QoS levels of A_x will be child nodes of every QoS level selected in A_y;
- Child nodes (i.e., QoS levels associated to an activity A_x) are sorted from left to right according to their utility values f_I. QoS levels with higher values of f_I are on the left and those with lower values are on the right.
- Add a virtual root node to all the nodes without incoming links.

Once the search tree is built, our heuristic algorithm ensures that its constituent QoS level compositions meet user QoS requirements. Towards this purpose, it first generates a global QoS aggregation formula (i.e., for the whole composition) for every QoS attribute by exploring the structure of the composition. Then it uses the generated formulas to compute the aggregated QoS value of each attribute and also the QoS



utility of every composition. The algorithm further checks the feasibility of these compositions by setting the global QoS constraints given by the user as upper bounds for the aggregated QoS values. The above step in performed along with the following optimizations aiming to prune the search tree of our algorithm.

a. *Pruning using incremental computation* As our algorithm traverses down the search tree from the root node to the leaf nodes, the aggregated QoS values increase along with the traversal of the tree. Consequently, if the aggregated QoS values calculated at any non-leaf node in the traversal of the tree, does not respect QoS constraints, then all the sub-tree under the non-leaf node will be pruned. This optimization is useful when we deal with long running processes having a large number of activities.

b. *Pruning using utility values approximation.* This idea concerns an approximation rather than an exact optimization. It utilizes the fact that our algorithm explores the search tree in an ordered way, i.e., it checks QoS levels with higher $\mathbf{f_l}$ values first. Therefore, if a QoS level $\mathbf{QL_l}$ does not lead to any feasible composition, all its following nodes (i.e., QoS levels of the same activity but with lower $\mathbf{f_l}$ values) will be not considered for the remainder computation, which reduces the number of services to check. This approximation is convenient when we have a large number of QoS levels per activity.

Our algorithm uses the above optimizations together, along with an additional improvement allowing to enhance the timeliness of the algorithm. Indeed, to reduce the time needed for computing the aggregated QoS values of QoS level compositions, we ensure that only one level changes when the algorithm switches from a composition to another. That is, the difference between two consecutive compositions $\mathbf{C_v}$ and $\mathbf{C_w}$ is that a QoS level $\mathbf{QL_i}$ in the first composition will be replaced by a service $\mathbf{QL_j}$ in the second one. Thus, instead of computing the whole aggregated QoS values of $\mathbf{C_w}$, the algorithm updates the aggregated QoS values of $\mathbf{C_v}$ with respect to $\mathbf{QoS_{cv}}$ and $\mathbf{QoS_{cw}}$.

Finally, our algorithm selects the QoS level composition that respects the user QoS requirements and yields the highest QoS utility $\mathbf{F}$. It further provides an executable and adaptive service composition underpinning dynamic binding of services. This composition is generated by binding every activity in the abstract composition to the set of services associated to the selected QoS level in the considered activity.

4.5. Experimental Evaluation

4.5.1. Experimental Setup

We conducted a set of experiments to evaluate the quality of our algorithm. These experiments were conducted on a Dell® machine with two AMD®Athlon 1.80GHz processors and 1.8 GB RAM. The machine is running under Windows XP® operating system and Java 1.6. In these experiments, we focus on three metrics:

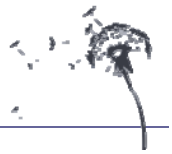
1. *Execution time*: It measures the response time of our algorithm with respect to the size of the problem in terms of the number of activities and the number of services per activity. In these experiments, we measure separately the execution time of local classification and global selection.

2. *Optimality:* This metric measures how close the composition given by our algorithm to the optimal composition (i.e., given by the brute-force algorithm) in terms of the obtained QoS utility. Therefore, the optimality metric is given by the following formula:

$$\text{Optimality} = \frac{F}{F_{Opt}}$$

where F is the utility of the composition given by our heuristic algorithm and F_{Opt} is the utility of the optimal composition given by the brute force algorithm.

3. *Accuracy*: The composition of QoS levels given by our algorithm respects global QoS requirements. Nevertheless, the service compositions yielded by combining the services associated to the selected QoS levels does not definitely ensure the global requirements since QoS of the centroids (i.e., QoS levels) is slightly different from QoS of services.



To measure the percentage of service compositions that does not respect global QoS requirements, we use the accuracy metric which is defined by the following formula:

$$\text{Accuracy} = \frac{n}{t}$$

where n is the number of service compositions that does not respect global QoS requirements and t is the total number of service compositions given by our heuristic algorithm. The parameter t is obtained by multiplying the number of services associated to each selected QoS level.

In our experiments, we use the data described in previous studies about Web Services' QoS [33,34]. In these studies, the authors provide a set of QoS metrics (i.e., response time, throughput, availability, validation accuracy, cost) related to current email validation Web services (Table 4-2). We use these metrics as a sample input data for our algorithm. Nevertheless, the number of Web services considered in these studies is too limited compared to the number of services that we need to assess the scalability of our algorithm. To this regard, we developed a Data Generator that randomly generates input data for our algorithm with respect to the range of values [min,max] where min and max denote respectively, the minimum and the maximum values of each QoS metric given in Table 4-2.

Service Provider	Response Time (ms)	Throughput (req./min)	Availability (%)	Validation Accuracy (%)	Cost (cents/invoke)
XMLLogic	720	6.00	85	87	1.2
XWebservices	1100	1.74	81	79	1
StrikeIron	912	10.00	96	94	7
CDYNE	910	11.00	90	91	2
Webservicex	1232	4.00	87	83	0
ServiceObjects	391	9.00	99	90	5

Table 4-2. QoS metrics for email validation Web Services.

Further, we developed a Process Generator that randomly generates abstract processes to use as input for experimenting with our algorithm. The Process Generator takes as arguments the number of activities and the number of candidate services per activity, and it yields as output a process by structuring the activities with respect to randomly chosen composition patterns. The Process Generator uses the Data Generator to provide the QoS values associated to service candidates of each activity in the process.

For the purpose of these experiments, we vary the number of activities and the number of services per activity between 10 and 50. Concerning the number of QoS constraints, it is comprised between 2 and 5 constraints. Finally, for the sake of precision we execute each experiment 10 times and we calculate the mean value of the obtained results.

Once data input is generated, we need to fix the values of the following parameters before launching the experiments:

- We set the values of global constraints given by the user to the mean value m of every QoS attribute aggregated with respect to the structure of the generated process composition. Thus, nearly 50% of the compositions will fulfill the user request.
- We use the method of computing QoS levels described in Section 4.4.3 to cluster service candidates according to a fixed number of clusters. This number is calculated by dividing the number of services candidates of every activity per 10. Thus, in theory, every cluster should include 10 service candidates.
- Concerning the computation of the utility threshold T, we fix it to m where m denotes the mean value of the utilities f_i of all the QoS levels associated to an activity.

4.5.2. Experimental Results

Figures 4-1 and 4-2 show the execution time of local classification and global selection respectively. These measurements are obtained by fixing the number of QoS constraints to 5 and varying the number of

activities between 10 and 50 and the number of service candidates per activity between 50 and 200. The obtained measurements show that the execution time of our algorithm increases in general along with the number of activities and services, as expected. Overall, in almost all cases our algorithm is executed in a reasonable amount of time (i.e., less than 0.5s) if we compare it, e.g., with the response time of the email validation Web services described in Table 4-2.

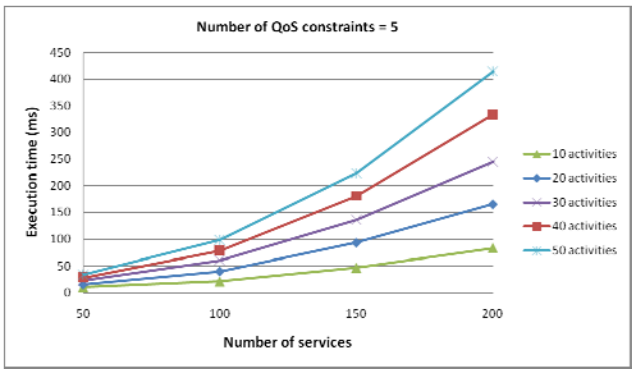


Figure 4-1. Execution time of the local classification phase

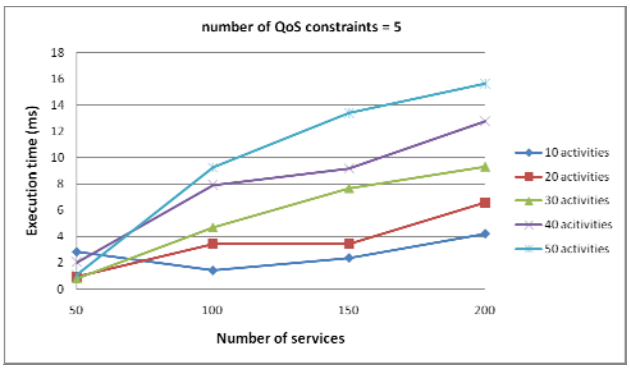


Figure 4-2. Execution time of the global selection phase

Concerning the optimality and the accuracy of our algorithm, we measure them while fixing the number of QoS constraints to 5, and varying the number of activities between 10 and 50 and the number of services per activity between 50 and 200.

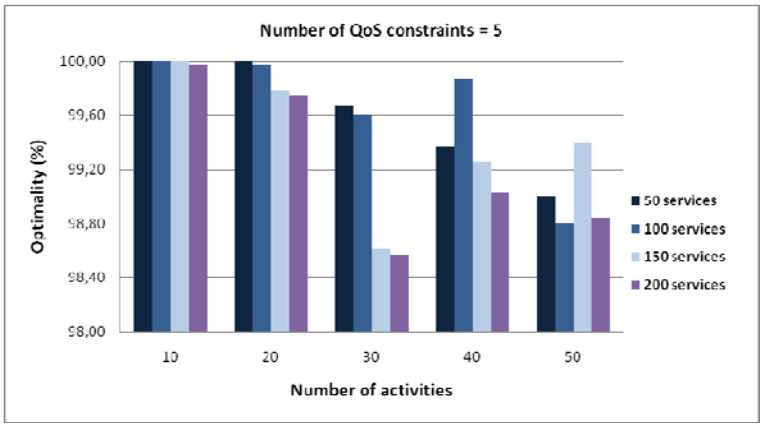


Figure 4-3. Optimality of our algorithm

As depicted in Figure 4-3, the optimality of our algorithm slightly decreases along with the number of activities or the number of services per activity. This means that, when it deals a large number of compositions, our algorithm discards more feasible compositions that may provide a better QoS utility, which is explained by the approximation introduced to prune the search tree of our algorithm (Section 4.4.4). Overall, our algorithm provides a satisfying QoS utility compared to the optimal compositions given the brute force algorithm (i.e., more than 98%).

Concerning the accuracy, Figure 4-4 shows that our algorithm gives also a satisfying accuracy (i.e., more than 98%). Regarding the quality of the composition given by our algorithm (i.e., according to the optimality metric), it is scarce that the services associated to the selected QoS levels do not respect the global QoS requirements, which explains the fact that the values of the accuracy remain approximately the same for different numbers of activities and services per activity.

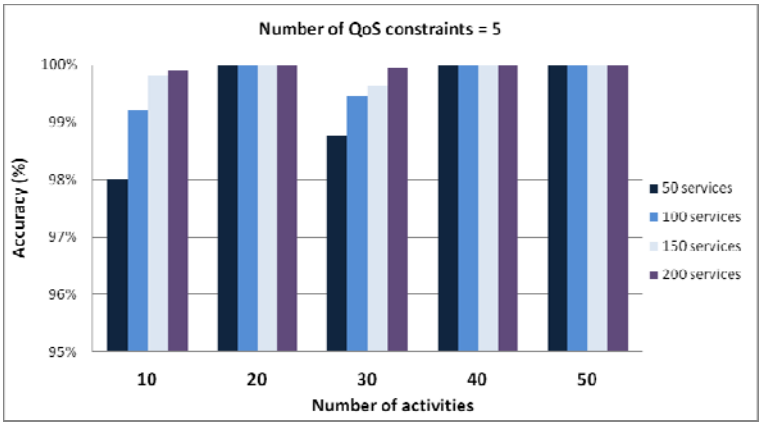


Figure 4-4. Accuracy of our algorithm

The optimality and the accuracy of our algorithm can be further enhanced by tuning the utility threshold **T** with respect to the density of services and the desired values of optimality and accuracy. Finally, it is worth noting that, during the experiments, the optimal service composition (i.e., service composition with the highest QoS utility) was always included in the set of service compositions provided by our algorithm.

.

5. Generation of executable user task

5.1. Extension Activity in Orchestra

A WS-BPEL process definition can include new activities, which are not defined by this specification, by placing them inside the <extensionActivity> element. These activities are known as extension activities. The deliverable T3.2D2 explains how to add new extension activities such as the late binding or monitoring Semeuse extension activities.

5.2. Late Binding Activity

SemEUsE specific BPEL 'extended activities' have been defined in order to deal with our late-binding goals: *lateBindingConfigure* and *lateBindingInvoke*, that are detailed in the two following sections. Their simultaneous use during BPEL execution is illustrated in Section 7.1.

5.2.1. lateBindingConfigure activity

The figure bellow gives the BPEL code for configuration of a late binding event that will take place between a BPEL process in need for a *getVideo* operation, and the two available services (*service1* and *service2*) able to provide it.

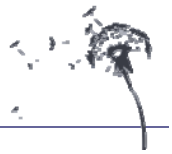
The late binding decision itself will take place in another activity (see 5.2.2), according to the common QoS properties required here in both the LCP-Net user preferences (*prefs.lcpnet*) [41] and web service agreements/contracts (*wsag1.xml* and *wsag2.xml*).

```
<semeuse:lateBindingConfigure invocationID="invocation1" preference="prefs.lcpnet" >
  <semeuse:candidateServices>
    <semeuse:service EPR="service1" portType="service1PT" operation="getVideo"
contract="wsag1.xml"/>
    <semeuse:service EPR="service2" portType="service2PT" operation="getVideo"
contract="wsag2.xml"/>
    (…)
  </semeuse:candidateServices>
</semeuse:lateBindingConfigure>
```

Technical information
Non-functional information

As such, the goal of the *lateBindingConfigure* activity is to configure, at runtime and **before effective service invocation**, all the components that will be necessary for service-binding decision making in the paired *lateBindingInvoke* activity.

Indeed, the monitoring component is one of them. For instance, through this extended activity, probes are deployed for active QoS monitoring on available Web services. Information used for probe deployment is extracted from both LCP-Nets preferences and WS-Agreements linked to the configuration activity. Examples are given bellow:



LCP-Net Preference Model (indicates **QoS dimensions** that need to be monitored)

```
<LCPnet:LCPnet name="Security_Camera_Preference" (...)>
  <nodes xsi:type="LCPnet:LNode" name="bandwidth" outArcs="(...)">
    <domain xsi:type="LCPnet:LNodeValue" name="Bandwidth_Low" linguisticValue="(...)"/>
    <domain xsi:type="LCPnet:LNodeValue" name="Bandwidth_Medium" linguisticValue="(...)"/>
    <domain xsi:type="LCPnet:LNodeValue" name="Bandwidth_High" linguisticValue="(...)"/>
    <domain xsi:type="LCPnet:CNodeValue" name="Bandwidth_measured_value"
crispValue="1.0"/>
    <linguisticTable>
      (...)
    </linguisticTable>
  </nodes>
  <nodes xsi:type="LCPnet:LNode" name="resolution" inArcs="(...)">
    <domain xsi:type="LCPnet:LNodeValue" name="Resolution_Low" linguisticValue="(...)"/>
    <domain xsi:type="LCPnet:LNodeValue" name="Resolution_High" linguisticValue="(...)"/>
    <domain xsi:type="LCPnet:CNodeValue" name="Resolution_measured_value"
crispValue="0.51"/>
    <linguisticTable>
      (...)
    </linguisticTable>
  </nodes>

  <arcs xsi:type="LCPnet:IArc" name="BtoS" startNode="//@nodes.1" endNode="//@nodes.0"/>
  <arcs name="BtoR" startNode="//@nodes.1" endNode="//@nodes.2"/>

  <valueDomains name="Bandwidth">
    <subsets name="Bandwidth_Low">
      <fuzzySubset y="1.0"/>
      <fuzzySubset x="0.5"/>
    </subsets>
    (...)
  </valueDomains>
  (...)
</LCPnet:LCPnet>
```

WS-Agreement (**technical information** for probe deployment)

```
<agreement:Agreement (...) >
  <name>agreement</name>
  <agreementId>id</agreementId>
  <context/>
  <terms>
    <all>
      <serviceDescriptionTerm name="getVideo" serviceName="getVideo" />
        <serviceProperties name="getVideo" serviceName="getVideo">
          <variableSet>
            <variable name="resolution">
              <location xsi:type="semeuse:WSBinding"
                    mode="pull" type="ws"
                    epr="http://localhost:8080/monitoring_firefighting/services/Camera1"
                    operation="getResolution" />
            </variable>
            <variable name="bandwidth">
              <location xsi:type="semeuse:WSBinding"
                    mode="pull" type="ws"
                    epr="http://localhost:8080/monitoring_firefighting/services/Camera1"
                    operation="getBandwidth" />
            </variable>
             (...)
            </variableSet>
        </serviceProperties>
    </all>
  </terms>
</agreement:Agreement>
```

The unique *invocationID* attribute is used for junction with the lateBindingInvoke activity.

5.2.2. lateBindingInvoke activity

The figure bellow gives the BPEL code for effective late binding invocation of services that where pinpointed in the paired lateBindingConfigure activity (see 5.2.1):

```
<extensionActivity>
    <semeuse:lateBindingInvoke invocationID="invocation1" inputVariable="invoke_in"
            outputVariable="invoke_out" preference="prefs.lcpnet"/>
</extensionActivity>
```

Technical information
Non-functional information

The goal of this extended activity is indeed to make both the late binding and invocation of an 'optimal' service during business process execution, by integrating non-functional user preferences (a.k.a. LCP-Nets) and current QoS values of services provided by the monitoring framework bundled into SemEUsE.

5.3. Security concerns

Security concerns are set in the XML companion file, depending on the service patterns attached to security preferences. As we implement the security service in a distributed way, the security manager has to extract the service security attributes so that the convenient security services can be executed (Figure 5-1). These attributes are then used to generate the convenient parameters in the XML service companion file (Figure 5-2).

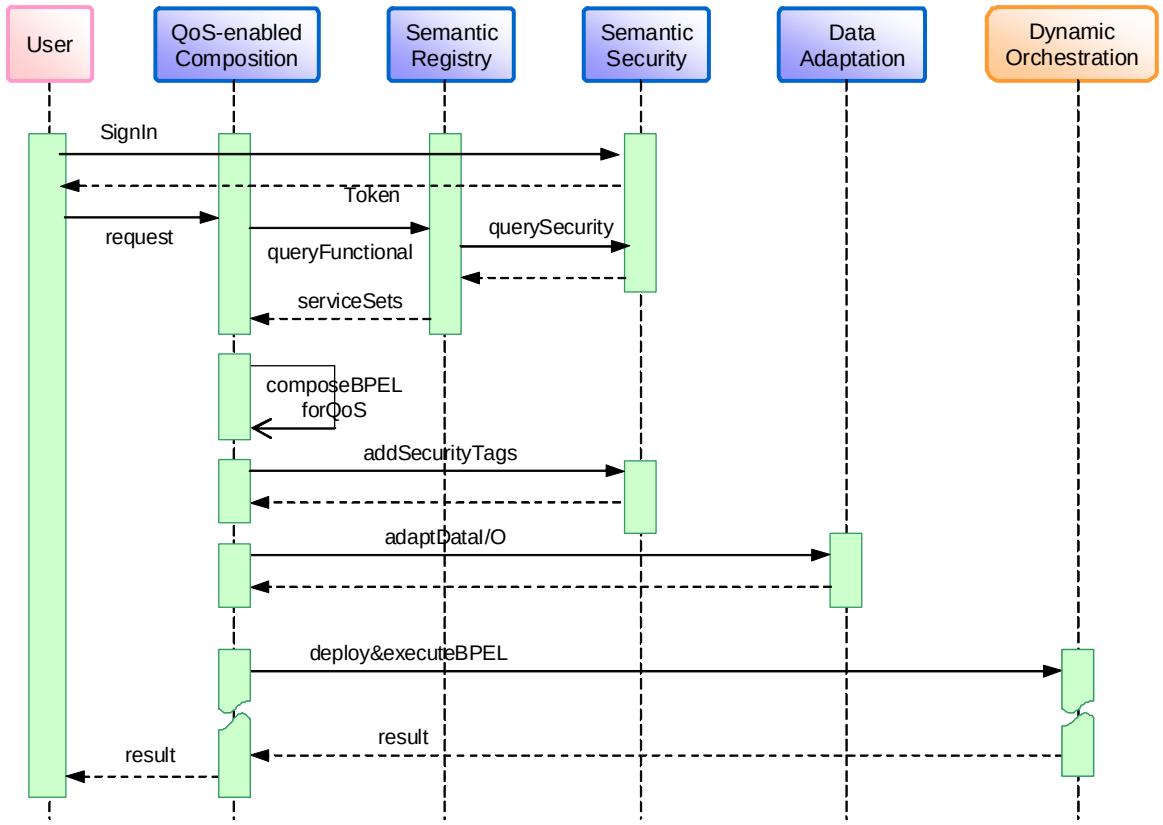
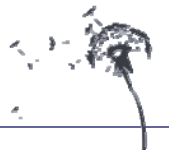


Figure 5-1: Security service integration in the QoS composition process

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/SecurityPreferences"
    xmlns:tns="http://xml.netbeans.org/schema/SecurityPreferences"
    elementFormDefault="qualified">
- <xsd:simpleType name="wssecurityType">
  <xsd:restriction base="xsd:anyURI" />
    </xsd:simpleType>
- <xsd:simpleType name="authenticationType">
  <xsd:restriction base="xsd:boolean" />
    </xsd:simpleType>
- <xsd:simpleType name="authorizationType">
  <xsd:restriction base="xsd:boolean" />
    </xsd:simpleType>
- <xsd:simpleType name="encryptionType">
  <xsd:restriction base="xsd:boolean" />
    </xsd:simpleType>
- <xsd:simpleType name="secureTransportType">
  <xsd:restriction base="xsd:boolean" />
    </xsd:simpleType>
- <xsd:simpleType name="digitalsignatureType">
  <xsd:restriction base="xsd:string" />
```

```
    </xsd:simpleType>
- <xsd:simpleType name="tokenType">
  <xsd:restriction base="xsd:string" />
    </xsd:simpleType>
- <xsd:complexType name="AccountType">
- <xsd:sequence>
  <xsd:element name="token" type="tns:tokenType" />
  <xsd:element name="subject" type="tns:subjectType" />
  <xsd:element name="principal" type="tns:PrincipalType" />
  <xsd:element name="IsPrincipal" type="tns:IsPrincipalType" />
    </xsd:sequence>
    </xsd:complexType>
- <xsd:simpleType name="uIdType">
  <xsd:restriction base="xsd:string" />
    </xsd:simpleType>
- <xsd:complexType name="subjectType">
- <xsd:sequence>
  <xsd:element name="uid" type="tns:uIdType" />
  <xsd:element name="domain" type="tns:domainType" />
    </xsd:sequence>
    </xsd:complexType>
- <xsd:complexType name="PrincipalType">
- <xsd:sequence>
  <xsd:element name="uid" type="tns:uIdType" />
  <xsd:element name="domain" type="tns:domainType" />
    </xsd:sequence>
    </xsd:complexType>
- <xsd:simpleType name="domainType">
  <xsd:restriction base="xsd:string" />
    </xsd:simpleType>
- <xsd:simpleType name="IsPrincipalType">
  <xsd:restriction base="xsd:boolean" />
    </xsd:simpleType>
- <xsd:complexType name="SecurityPreferencesType">
- <xsd:sequence>
  <xsd:element name="account" type="tns:AccountType" />
- <xsd:element name="requirements">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="authentication" type="tns:authenticationType" />
  <xsd:element name="authorization" type="tns:authorizationType" />
  <xsd:element name="secureTransport" type="tns:secureTransportType" />
  <xsd:element name="encryption" type="tns:encryptionType" />
  <xsd:element name="signature" type="tns:digitalsignatureType" />
    </xsd:sequence>
    </xsd:complexType>
    </xsd:element>
  <xsd:element name="wssecurity" type="tns:wssecurityType" />
    </xsd:sequence>
    </xsd:complexType>
  <xsd:element name="securityPreferences" type="tns:SecurityPreferencesType" />
    </xsd:schema>
```

Figure 5-2: XML security preference parameter file



5.4. Monitoring Activity

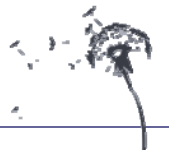
A SemEUsE specific BPEL 'extended activity' has been defined in order to deal with the activation and deactivation of the monitoring for Late Binding. The figure bellow shows the syntax of this *monitoring* activity.

```
<extensionActivity>
    <semeuse:monitoring state="start|stop" />
</extensionActivity>
```

We distinguish two monitoring commands for this activity:

- Monitoring start (*state="start"*): activate *Views*, QoS data access points of M4LB (i.e. the Monitoring for Late Binding, cf. T3.2D1 deliverable), that deploys probes and starts to collect, process, and cache QoS data for available Web services. As such: (i) before this command is called, all the service properties that need to be monitored should be specified (see section lateBindingConfigure activity, where the *lateBindingConfigure* activity provides the necessary information including candidate Web services and their associated WS-Agreements) ; (ii) after this command is called, the late service binding decision can take place (see Section 5.2.2, where the *lateBindingInvoke* activity selects an 'optimal' service during business process execution, by querying current QoS values of services provided by M4LB).
- Monitoring stop (*state="stop"*): deactivate *Views*, i.e. stops to collect QoS data and undeploys probes of Web services that have been processed in the *monitoring start* command.

Detailed information about the M4LB contribution can be found in the WP3 T3.2D1 deliverable.



6. Deployment of executable user task

6.1. User task BPEL deployment

Once designed, the BPEL process can be deployed on PEtALS. The way how to deploy a BPEL process and create a new Service endpoint on the bus is described in the T3.2D2 internal deliverable.

6.2. SLA deployment

PEtALS contains static probes deployed on its internal components such as Delivery Channels and Normalized Message Routers (NMR) to monitor its behavior. When a provider wants to install a service on the bus, it must configure a Service Unit (SU) in “provide” mode. In this SU, it can decide to active a dynamic probe. This dynamic probe is in charge to monitor the measurable parameters defined in the SU (response time, availability, etc.). These parameters correspond to the WSDM specification.

When Dragon is synchronized with PEtALS' technical registry, it is able to automatically generate the service terms of an agreement because it precisely corresponds to measurable and exposed properties associated with a service provider.

Once guarantee terms (e.g. SLOs, penalties and/or rewards) are defined by the client and the provider, the agreement can be signed by these two parts. This signature starts the deployment process.

The client's SU (in “consume mode”) is generated using the guarantees terms which are needed to configure the dynamic probes. This SU activates the client endpoint. A specific connection between client and provider is now established. This way, PEtALS can monitor all the SLOs defined in the agreement signed between these two parts thanks to the probes deployed on all the components used in this connection. This SLA deployment process is explained, in details, in deliverable T3.2D2.

Last, the monitoring capability, proposed by EBM WebSourcing and France Télécom, also includes a Service Level Checking (SLC) part. This part is based on data provided by PEtALS. This data is related to the connections established between the clients and the service providers. The SLC's goal is to check if the agreements (i.e., the SLAs) defined between the clients and the service providers are respected or violated. The SLC's results are displayed via an IHM designed and developed by INRIA Tuvalu. Detailed information about the SLC contribution can be found in the WP3 T3.1D2 deliverable.

6.3. Late Binding Monitoring deployment

The aim of the late-binding approach is to establish pairings between a process and services on the fly. It cares about selecting the best services according to their current QoS, including domain-specific and technical QoS indicators collected from probes associated to the candidate services. QoS measurement is implemented by a specific component: M4LB (Monitoring for Late Binding, T3.2D1 deliverable) that allows for subscribing and querying monitoring data associated to candidate services during business process execution. The integration of the late binding decision framework (T3.2D3 deliverable) in a QoS aware BPEL engine (T3.2D2 deliverable) relies on the Orchestra WS-BPEL 2.0 orchestration engine which allows extending activities with late binding capabilities.

M4LB is co-localized with the late binding component. It collects, processes, and caches instantaneous QoS data during process execution, providing solutions for requirements in monitoring: flexibility as well as performance in data access for clients, coherency of data sets and optimisations in network usage. The communication between M4LB and data sources allows for collecting i) technical QoS data (e.g., response time) from the monitoring framework integrated in PETALS, and ii) domain-specific QoS data (e.g., cardiac rhythm of firemen, as proposed in the SemEUsE fire-fighting use case) either from service endpoints or from a dedicated data collector application. The configuration of the communication channels relies on Service property information found in the WS-Agreement documents attached to the candidate services. The configuration of M4LB by itself relies on a view description document referring WS-Agreement documents. Detailed information about the M4LB contribution can be found in the WP3 T3.2D1 deliverable.

7. User task execution

7.1. Late binding of services

7.1.1. The need for late binding of services to processes

Service-oriented architectures (SOA) deal with the growing need for distributed applications capable of evolving continuously over their execution.

In the context of the web, services can appear and disappear at runtime, thus requiring a particularly loose coupling between service providers and consumers. Also failure or disconnections are likely, especially in the context of pervasive SOA. Variations in the QoS of services are also a concern, given the very large base of equivalent service offers.

Indeed, to cope with the dynamism of the Web and get this loose coupling, the binding of Web services (from providers) to business processes (of consumers) should be established on the fly, at runtime. In fact, only then could high interoperability among heterogeneous service offers and requests could be provided.

As such, a dedicated late binding component is used with our decision engine (cf. SemEUsE T3.2D3 document) to *decide* at runtime which service will be called to satisfy each consumers' non-functional requirements (including its preferences), taking into account the latest QoS values of available services. In this context, the available monitoring middleware allows us to dynamically query services in order to efficiently obtain their current QoS values.

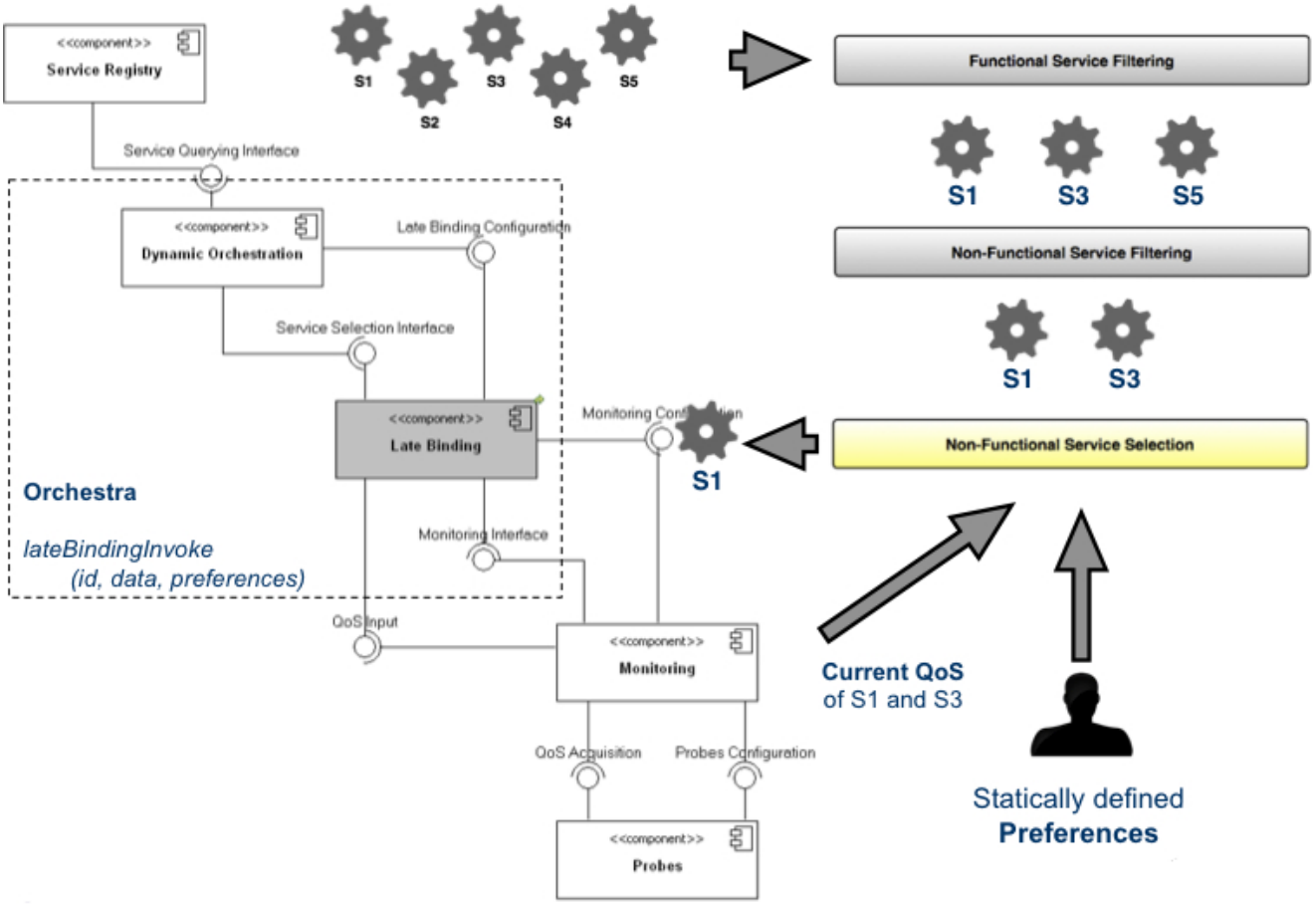


Figure 7-1 - Late binding component and process

7.1.2. Implementation in SemEUsE: BPEL execution

Orchestra has been chosen as the BPEL orchestration engine by the consortium. We add SemEUsE specific 'extended activities' processing abilities to the engine in order to deal with our late-binding goals: *lateBindingConfigure* and *lateBindingInvoke* (as seen in 5.2.1 and 5.2.2). The first one, located at the very

beginning of a BPEL business process allows for the configuration of late binding requirements (i.e. monitoring framework) before the service invocation itself is processed; the second one is where the late service binding decision takes places.

A high-level view of the late binding process, during BPEL execution, where 'configuration' and 'invocation' are collapsed for easy understanding is given in Figure 7-1. In this figure the various SemEUsE components involved in the filtering, selection and late binding of services are shown on the left-hand side (service registry, dynamic orchestration, late-binding, monitoring and probes), while the logical steps undertaken for invocation are shown on the right (they include, in their respective usage order, functional service filtering, non-functional service filtering and non-functional service selection).

Non-functional service selection is the main step we are interested about, it is the logical view including both the *lateBindingConfigure lateBindingInvoke* and extended activities. As such, it has decision inputs the current QoS values provided by the Monitoring component developed by Orange Labs and the LCP-Nets user preferences concerning this specific late binding invocation In fact, given a static LCP-Net preference model, it will be translated into an efficient and dynamic decision framework [42]:

- Utility tables transformed into fuzzy rules sets, as in Fuzzy Control.
- System inputs are current QoS values of selected services, for each monitored dimension.
- System outputs are each service utility, given as a crisp (e.g. 3.0) or linguistic value (e.g. high).

Here is a small BPEL example, where both activities, as seen in 5.2.1 and 5.2.2, are used in the place of the standard invoke:

```
<process name="semeuse" (...) >
  <!-- (...) Import client WSDL, define partner links, define variables -->

  <sequence name="main">
    <extensionActivity>
      <semeuse:lateBindingConfigure invocationID="invocation1"
preference="prefs.lcpnet">
        <semeuse:candidateServices>
          <semeuse:service EPR="service1" portType="service1PT" operation="getVideo"
                    contract="wsag1.xml"/>
          <semeuse:service EPR="service2" portType="service2PT" operation="getVideo"
                    contract="wsag2.xml"/>
        </semeuse:candidateServices>
      </semeuse:lateBindingConfigure>
    </extensionActivity>

    <extensionActivity>
      <semeuse:monitoring state="start" />
    </extensionActivity>

    <!-- (...) Receive input from requester. -->

    <extensionActivity>
     <semeuse:lateBindingInvoke invocationID="invocation1" inputVariable="invoke_in"
                    outputVariable="invoke_out" preference="prefs.lcpnet"/>
    </extensionActivity>

        <!-- (...) Generate reply to synchronous request -->
    <extensionActivity>
      <semeuse:monitoring state="stop" />
    </extensionActivity>
  </sequence>
</process>
```

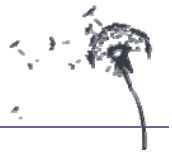
7.2. Data I/O adaptation

7.2.1. Introduction

In a service-oriented architecture where services are statically linked one to another, communication between them is an issue. Many often, it is difficult to predict the target service receiving data from yours, hence you need some kind of data adaptation process that will format your data to the one expected by the receiving service. The same goes for the returned response from the invoked service. This process ensures the interaction and interoperability of services that are not statically designed to communicate with each others and have no knowledge of any data formats but their own.

This is certainly not a trivial issue but the SemEUsE data adaptation component will strive to give it a satisfying response. Semantic is the magic powder that makes data adaptation more successful. The data adaptation algorithm heavily relies on an ontology that structures the knowledge base of the domain and defines relationships between concepts. These relationships are exploited during the process by the reasoning engine.

Obviously, the algorithm can only perform when the services that it is dealing with have their data types semantically annotated i.e. there exists a link between a data type and an ontology concept.



7.2.2. Semantic annotation

This paragraph presents how to semantically annotate the data types.

Data types are typically defined in XML schemas (XSD). Semantic annotations are inserted into XML schemas as attributes, i.e. `sawsdl:modelReference`, to schema elements. The value of the attribute is a URI that points to a concept of an ontology shared by both the invoker and the invoked services. This concept defines the *type* of the element. A second concept can be added (and therefore a second URI) to define the *use* of the element.

This is an example of a semantically annotated XSD (SA-XSD). It shows three annotated elements: *firstname*, *surname* and *age*. Each of them is linked with two concepts from the ontology.

```
<xs:schema version="1.0"
  targetNamespace="http://com.thalesgroup.setha.tutorial.sampleservices.s1"
  xmlns:tns="http://com.thalesgroup.setha.tutorial.sampleservices.s1"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="client" type="tns:clientType" />
  <xs:complexType name="clientType">
    <xs:sequence>

      <xs:element name="firstname" type="xs:string"
sawsdl:modelReference="http://www.example.com/semeuse.owl#first_name!type
http://www.example.com/semeuse.owl#client!utility" />

      <xs:element name="surname" type="xs:string"
sawsdl:modelReference="http://www.example.com/semeuse.owl#family_name!type
http://www.example.com/semeuse.owl#client!utility" />

      <xs:element name="other_info" type="tns:otherinfoType" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="otherinfoType">
    <xs:sequence>

      <xs:element name="age" type="xs:int"
sawsdl:modelReference="http://www.example.com/semeuse.owl#age!type
http://www.example.com/semeuse.owl#client!utility" />

    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

7.2.3. Processing the annotated input data

In order to ease the use and the integration of the data adaptation component in various environments, it is assumed that the payload that needs to be adapted to a target format is flat. Why such an assumption? Let us look at the task of someone who is designing a BPEL process. At design time, if we were to ask him to set the input data in a particular format, which format would he choose knowing that he has absolutely no clue about whether this format would match the one expected by the selected service? The simplest solution is to let him set the input data in no particular format, i.e., no order and no hierarchy, this is the flat format (see Figure 7-2).

In the case the designer of the BPEL process decides to structure the input data, the current implementation will flatten the structure.

7.2.4. Algorithm

The data adaptation component receives as input the selected service and the input data to ground. At this stage, it is assumed that a pre-grounding phase is already executed and the grounding would succeed. The adaptation algorithm will not check it again.

The only verification performed comes from the constraints described below. They are meant to lessen the risks of indecision during the adaptation. This data adaptation algorithm does not pretend to solve every situation and these restrictions are to ensure that the grounding will be achieved:

- Each element of a XML schema with no children has to be annotated.

- There shall not be two elements, from the target data format, annotated with *compatible* concepts (i.e. concepts that are not linked with an *is_a* or *is_equivalent* relationship) as it would make it very difficult for the algorithm to decide which element to favour.
- A schema that defines nested list is not resolvable.

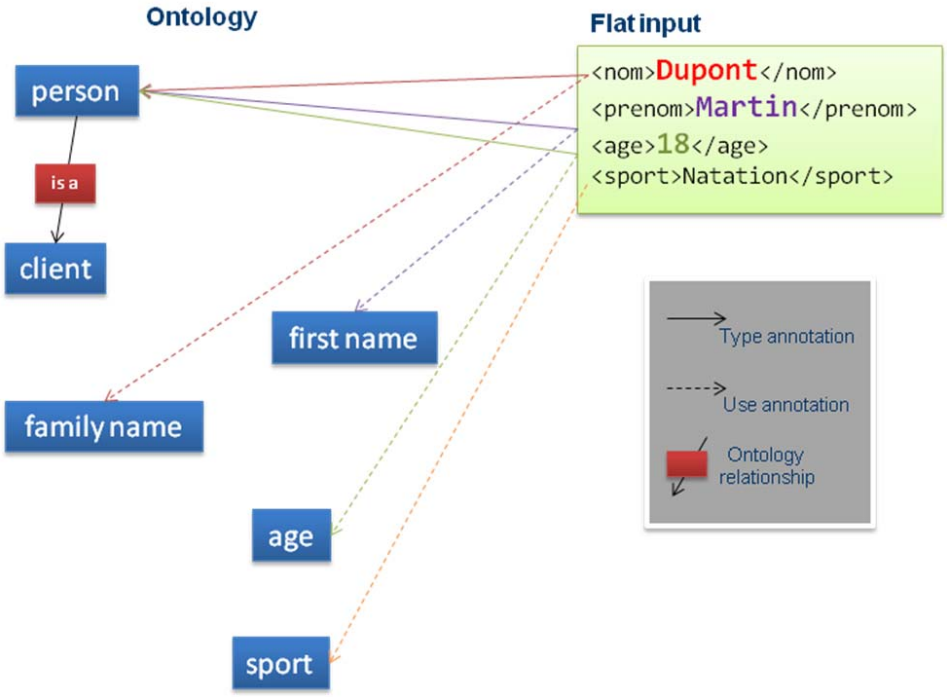


Figure 7-2: Semantically annotated payload

Figure 7-3 depicts an example of the result of the data adaptation given the input from the BPEL process and the capacity described by the SA-XSD previously described.

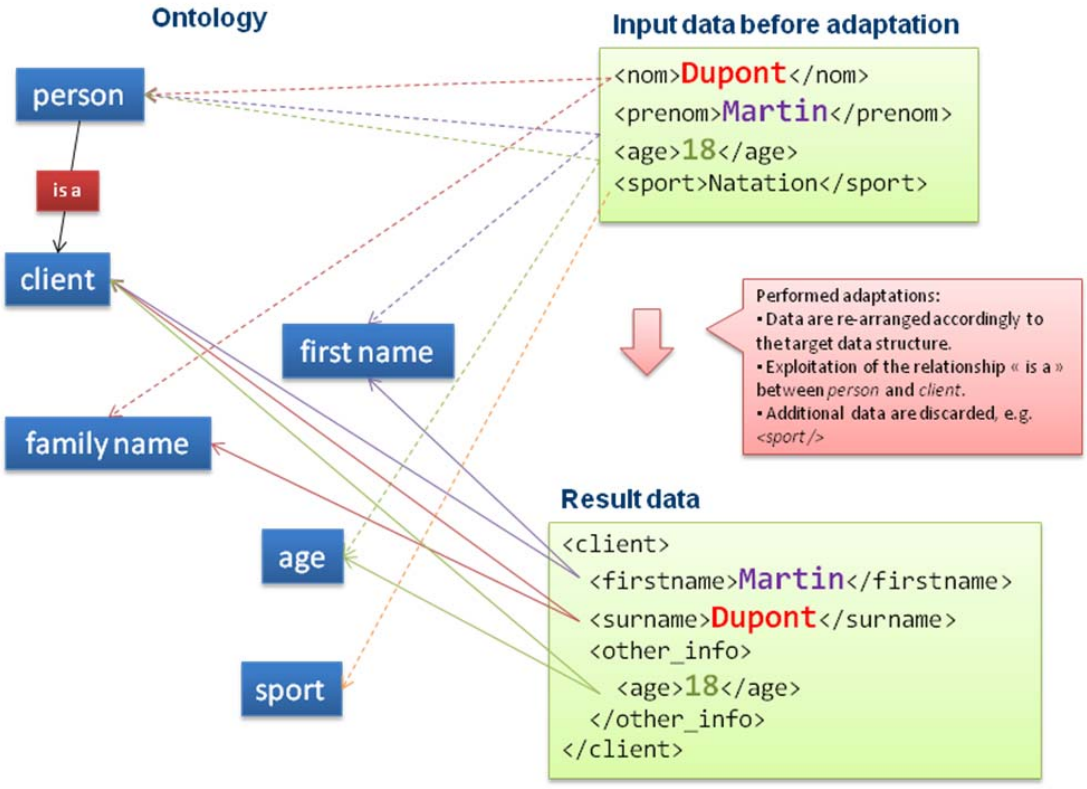


Figure 7-3: Example of data adaptation

7.2.5. JBI connection

After having grounded the input, the data adaptation component proceeds with the actual invocation of the service. The component generates a NMR message to forward payload from the BPEL process to selected service through PEtALS ESB. It then decodes the response NMR message coming from the service, grounds it and forwards the message back to the BPEL process.

7.3. Late security binding

In order to fulfil legal requirements, service invocation and the different authorisation information must be logged at the runtime. To fit this objective, the late-binding system has to invoke the security mediator. The parameters associated to the required patterns are extracted from the XML file (Figure 5-2), so that the credential manager can be invoked before setting the different security tags that will be used at the service layer to select the requested components accordingly (namely the message signature and the transport system) (Figure 7-4).

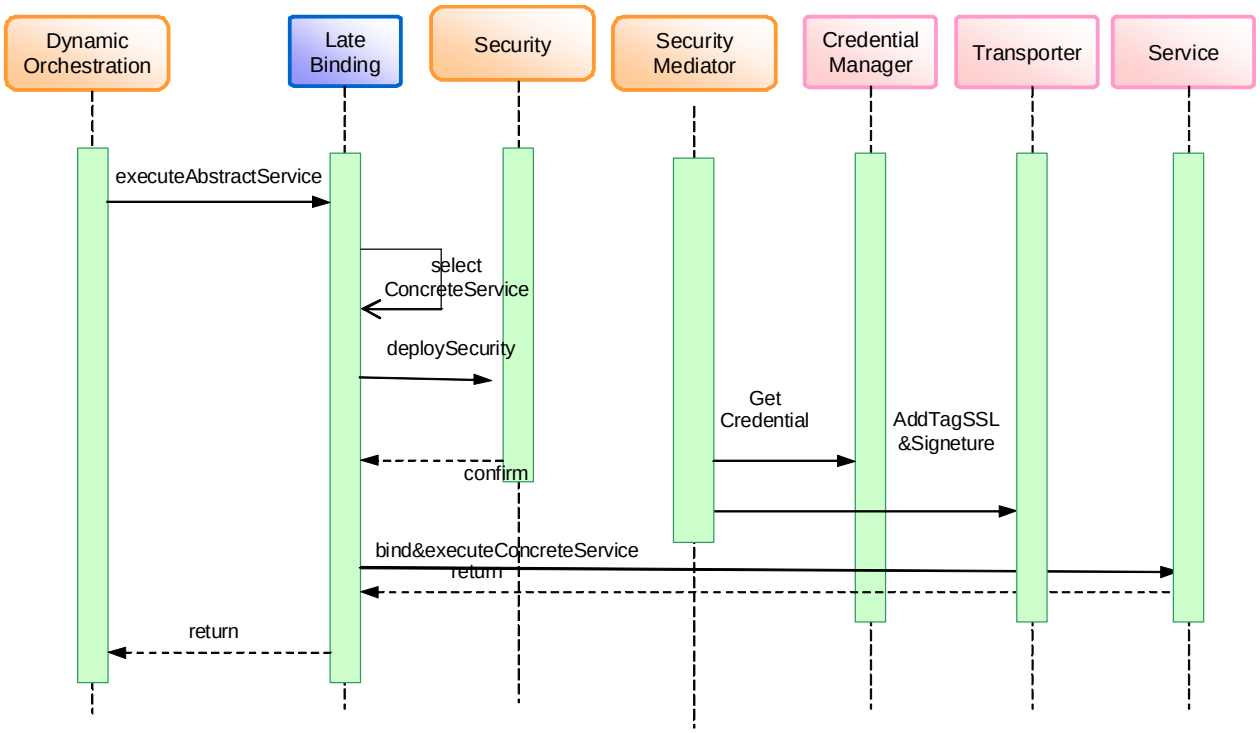
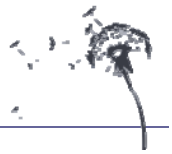
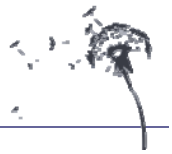


Figure 7-4: Invocation of the security service in the late-binding component



8. Conclusion

This deliverable presented in a pretty comprehensive way the whole service composition approach within SemEUsE, referencing in addition, where appropriate, other deliverables that complement the present document. This approach provides a powerful solution to automated, semantic, dynamic, just-in-time, end-to-end QoS- and security-aware, runtime monitored, and data I/O adapted composition of services for fulfilling a user request.



9. References

[1] L. Capra, S. Zachariadis, and C. Mascolo. Q-CAD: QoS and Context Aware Discovery Framework for Mobile Systems. In ICPS, page 453-456, 2005.

[2] [Dobson Lock & Sommerville, 2005] Dobson, G.; Lock, R. & Sommerville, I. QoSOnt: a QoS Ontology for Service-Centric Systems 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005, 80-87

[3] G. Dobson and A. Sanchez-Macian, “Towards Unified QoS/SLA Ontologies,” in SCW ’06: Proceedings of the IEEE Services Computing Workshops. Washington, DC, USA, 2006, 169–174.

[4] J. R. Hobbs and F. Pan. Time Ontology in OWL, 2006. http://www.w3.org/TR/owl-time/.

[5] S. Kalasapur, M. Kumar, and B. Shirazi. Evaluating Service Oriented Architectures (SOA) in Pervasive Computing. In PERCOM ’06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, pages 276–285, Washington, DC, USA, 2006. IEEE Computer Society.

[6] [Kim Sengupta & Evermann 2005] H. M. Kim, A. Sengupta, and J. Evermann, “MOQ: Web Services Ontologies for QoS and General Quality Evaluations,” in European Conference on Information Systems (ECIS 2005), 2005.

[7] C. Marchetti, B. Pernici, and P. Plebani. A Quality Model for Multichannel Adaptive Information. In WWW Alt. ’04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, pages 48–54, New York, NY, USA, 2004. ACM.

[8] Maximilien, E. & Singh, M. A framework and ontology for dynamic Web services selection Internet Computing, IEEE, 2004, 8, 84-93.

[9] L. Mcnamara, C. Mascolo, and L. Capra. Trust and Mobility Aware Service Provision for Pervasive Computing. In First International Workshop on Requirements and Solutions for Pervasive Software Infrastructures, 2006.

[10] D. Min, E. Kim, Y. Lee, G. Kang, and J. B. Clark, “Web Services Quality Model,” 2007, http://www.oasisopen.org/committees/tc home.php?wg abbrev=wsqm.

[11] E. D. Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl. A Journey to Highly Dynamic, Self-Adaptive Service-Based Applications. Automated Software Eng., 15(3-4):313–341, 2008.

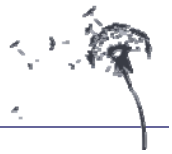
[12] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour. Semantic WS-Agreement Partner Selection. In WWW ’06: Proceedings of the 15th international conference on World Wide Web, pages 697–706, New York, NY, USA, 2006. ACM.

[13] I. V. Papaioannou, D. T. Tsesmetzis, I. G. Roussaki, and M. E. Anagnostou, “A QoS Ontology Language for Web-Services,” in AINA ’06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications. Washington, DC, USA, 2006, 101–106.

[14] G. Resta and P. Santi. WiQoSM: An Integrated QoS-Aware Mobility and User Behavior Model for Wireless Data Networks. IEEE Transactions on Mobile Computing, 7(2):187–198, 2008.

[15] K. Tserpes, D. Kyriazis, A. Menychtas, T. A. Varvarigou, F. Silvestri, and D. Laforenza. An Open Architecture for QoS Information in Business Grids. In T. Priol and M. Vanneschi, editors, CoreGRID, pages 37–49. Springer, 2007.

[16] Michael C. Jaeger. Modelling of Service Compositions: Relations to Business Process and Workflow Modelling. In 1st International Workshop on Service-Oriented Modelling., pages 14{25, Chicago, Illinois, USA, December 2006.



[17] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing: State of the Art and Research Challenges. Computer, 40(11):38-45, 2007.

[18] Cesare Pautasso and Gustavo Alonso. Flexible Binding for Reusable Composition of Web Services. In Proc. of the 4th Workshop on Software Composition (SC 2005), Edinburgh, Scotland, April 2005.

[19] Massimiliano Di Penta, Ra_aele Esposito, Maria Luisa Villani, Roberto Codato, Massimiliano Colombo, and Elisabetta Di Nitto. WS Binder: A Framework to Enable Dynamic Binding of Composite Web Services. In SOSE '06: Proceedings of the 2006 international workshop on Service-oriented software engineering, pages 74-80, New York, NY, USA, 2006. ACM.

[20] Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. QoS-Aware Middleware for Web Services Composition. IEEE Trans. Softw. Eng., 30(5):311-327, 2004.

[21] Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. ACM Trans. Web, 1(1):6, 2007.

[22] Michael C. Jaeger and Gero Mühl. QoS-based Selection of Services: The Implementation of a Genetic Algorithm. In Torsten Braun, Georg Carle, and Burkhard Stiller, editors, Kommunikation in Verteilten Systemen (KiVS 2007) Industriebetrage, Kurzbeiträge und Workshops, pages 359-350, Bern, Switzerland,
March 2007. VDE Verlag, Berlin und Offenbach.

[23] Ziad Kobti andWang Zhiyang. An Adaptive Approach for QoS-Aware Web Service Composition Using Cultural Algorithms. In Mehmet A. Orgun and John Thornton, editors, Australian Conference on Artificial Intelligence, volume 4830 of Lecture Notes in Computer Science, pages 140-149. Springer, 2007.

[24] Mohammad Alrifai, Thomas Risse, Peter Dolog, and Wolfgang Nejdl. A Scalable Approach for QoS-based Web Service Selection. In 1st International Workshop on Quality-of-Service Concerns in Service Oriented Architectures (QoSCSOA'08) in conjunction with ICSOC 2008, Sydney, December 2008.

[25] Sonia Ben Mokhtar, Anupam Kaul, Nikolaos Georgantas, and Valérie Issarny. Efficient semantic service discovery in pervasive computing environments. In Middleware '06: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware, pages 240-259, New York, NY, USA, 2006. Springer-Verlag New York, Inc.

[26] Sonia Ben Mokhtar, Davy Preuveneers, Nikolaos Georgantas, Valérie Issarny, and Yolande Berbers. EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. J. Syst. Softw., 81(5):785-808, 2008.

[27] S.P. Lloyd. Least Squares Quantization in PCM. Unpublished Memorandum, Bell Laboratories, 1957.

[28] Nebil Ben Mabrouk, Nikolaos Georgantas, and Valérie Issarny. A Semantic End-to-End QoS Model for Dynamic Service Oriented Environments. In Principles of Engineering Service Oriented Systems (PESOS'09), held in conjunction with the International Conference on Software Engineering (ICSE'09), 2009.

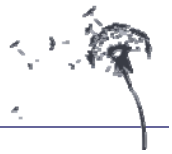
[29] Francesco Moscato, Nicola Mazzocca, Valeria Vittorini, Giusy Di Lorenzo, Paola Mosca, and Massimo Magaldi. Workow Pattern Analysis in Web Services Orchestration: The BPEL4WS Example. In High Performance Computing and Communcations, pages 395-400, 2005.

[30] Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, and Arthur. Pattern Based Analysis of BPEL4WS. In Conceptual Modeling - ER 2003, pages 200-215, 2003.

[31] David Arthur and Sergei Vassilvitskii. On the Worst Case Complexity of the kmeans Method. Technical Report 2005-34, Stanford InfoLab, 2005.

[32] Leslie Hogben, Anne Greenbaum, Richard Brualdi, and Roy Mathias. Handbook of Linear Algebra. Chapman & Hall, January 2007.

[33] E. Al-Masri and Q.H. Mahmoud. QoS-based Discovery and Ranking of Web Services. pages 529-534, August 2007.



[34] Eyhab Al-Masri and Qusay H. Mahmoud. Discovering the Best Web Service. In WWW '07: Proceedings of the 16th international conference on World Wide Web, pages 1257-1258, New York, NY, USA, 2007. ACM.

[35] John A. Rice. Mathematical Statistics and Data Analysis. Duxbury Press, April 2001.

[36] Alberts C., Dorofee A., 2001. An Introduction to the OCTAVE[SM] Method. CERT White paper available at http://www.cert.org/octave/methodintro.html

[37] Andrieux A.; Czajkowski K.; Dan A.; Keahey K.; Ludwig H.; Pruyne J.; Rofrano J.; Tuecke S. ; Xu, 2007, Web Services Agreement Specification (WS-Agreement) available at: http://www.ogf.org/documents/GFD.107.pdf

[38] Argonne National laboratory, WS-Agreement Structure, Version 0.1, 2004 available at: http://www.mcs.anl.gov/~keahey/Meetings/GRAAP/WS-Agreement%20Structure.pdf

[39] F. Biennier, X. Boucher, A. Hammami, L. Vincent. Towards a modelling framework for networks of SMEs. Actes de la conférence PRO-VE'02. Sesimbra (Portugal) Mai 2002. In Collaborative business ecosystems and virtual enterprises, Camarinha-Matos L. Ed. Kluwer academic publishers. pp. 11- 18

[40] Chen T.Y., Chen Y.M., WangC.B., Chu H.C., Yang H., 2007. Secure resource sharing on cross-organization using a novel trust method. Robotics and Computer-Integrated Manufacturing Vol. 23, pp. 421-435.

[41] Pierre Châtel, Isis Truck, Jacques Malenfant. A linguistic approach for non-functional constraints in a semantic SOA environment. in FLINS'08. 2008. Madrid, España.

[42] Pierre Châtel, Isis Truck, Jacques Malenfant, LCP-Nets: A linguistic approach for non-functional preferences in a semantic SOA environment. Journal of Universal Computer Sciences (JUCS), 2009(Special Issue on Information Fusion and Logic-based Reasoning Approaches for Decision Making Under Uncertainty).

[43] CLUSIF, 2000. Mehari. 91p, available at <https //www.clusif.asso.fr/fr/production/ouvrages/pdf/MEHARI.pdf>

[44] Covington J.M., Moyer J.M., Mustaque A., 2001. Generalized Role-Based Access Control for Securing Future Applications, Proc. (ICDCS'01), pp. 391-401.

[45] Common criteria organisation, 2000. Common criteria an introduction. Available at http://www.commoncriteria.org/introductory_overviews/CCIntroduction.pdf, 20 p.

[46] DCSSI (Direction centrale de la sécurité des systèmes d'information), 2004. Expression des Besoins et Identification des Objectifs de Sécurité : EBIOS, Rapport Technique. Available at <http://www.ssi.gouv.fr/fr/confiance/ebios.html>

[47] Department Of Defense (DoD), 1985. Trusted Computer Security Evaluation Criteria - Orange Book. DOD 5200.28-STD report

[48] Department of Defense , 2005. *NRL Security Ontology*, available at: http://chacs.nrl.navy.mil/projects/4SEA/ontology.html

[49] EEC, 1991. Information Technology Security Evaluation Criteria. Zip file downlable at http://www.cordis.lu/infosec/src/crit.htm

[50] Ferraiolo D., Cugini J., KuhnR., 1995. Role Based Access Control: Features and Motivations, Proc. Annual Computer Security Applications Conference, pp. 554-563.

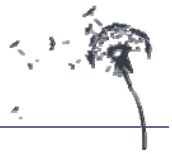
[51] ISO, 2000. ISO/IEC 17799:2000 standard - Information technology. Code of practice for information security management.

[52] ISO, 2001. ISO/IEC 9594-1. Information Technology – Open Systems Interconnection – The Directory: overview of concepts, models and services. 23p.

[53] Lin A., Brown R., 2000. The application of security policy to role-based access control and the common data security architecture. Computer Communications, Vol.23, pp. 1584-1593.

[54] Li D., Hu S., Bai S., 2002. A uniform model for authorization and access control in enterprise information platform. Proc. EDCIS2002, Lecture Notes in Computer Science, vol. 2480, pp.180-192.

[55] Moore, A. P., Ellison, R. J., 2001. Architectural Refinement for the Design of Survivable Systems. Technical Note SOFT01(CMU/SEI-2001-TN-008). Software Engineering Institute, Carnegie Mellon University. Octobre 2001. Disponible at: http://www.sei.cmu.edu/publications/documents/01.reports/01tn008.html



[56] OASIS, Organization for the Advancement of Structured Information Standards, 2002. Security Assertion Markup Language (SAML), available at: xml.coverpages.org/saml.html

[57] Organization for the Advancement of Structured Information Standards (OASIS), 2003. eXtensible Access Control Markup Language (XACML) TC, available at: www.oasisopen.org/committees/tc_home.php?wg_abbrev=xacml

[58] OASIS Service oriented architecture reference model TC, 2008. Reference architecture for service oriented architecture. Version 1.0. 104 Pages. Available online at http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf

[59] Ray I., Kumar M., Lijun, Y., 2006. LRBAC: A Location-Aware Role-Based Access Control, Proc.ICISS2006, Lecture Notes in Computer Science Vol. 4332, pp. 147-161.

[60] Wainer J., Kumar A, Barthelmess P., 2007. DW-RBAC: A formal security model of delegation and revocation in workflow systems, Information Systems, Vol. 32, pp. 365-384.

[61] Weber B., Reichert M., Wild W., Rinderle S., 2005. Balancing Flexibility and Security in Adaptive Process Management, proc. CoopIS, DOA, and ODBASE 2005, Lecture Notes in Computer Science, Vol. 3760, pp. 59-76.

[62] Wang X., Zhang Y., Shi H., Yang J., 2008. BPEL4RBAC: An Authorisation Specification for WS-BPEL. in J. Bailey et al. (Eds.): WISE 2008, LNCS 5175, pp. 381–395,

[63] Xiangpeng Z., Cerone A., Krishnan P., 2006. Verifying BPEL Workflows Under Authorisation constraints. In S. Dustdar, J.L. Fiadeiro, and A. Sheth (Eds.): BPM 2006, LNCS 4102, pp. 439–444,

[64] Yialelis N., Lupu E., Sloman M., 1996. Role-Based Security for Distributed Object Systems, Proc. ICE'96, pp. 80–85.

[65] Zhang G., Parashar M., 2006. Dynamic context aware access control for grid application, Proc. of GRID'XY, Computers & Security, Volume 25, pp. 507-521.

[66] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification. Open Grid Forum (OGF), Grid Resource Allocation Agreement Protocol (GRAAP) WG, March 2007. http://forge.gridforum.org/sf/projects/graap-wg.